A New Iterative Method for Solving Nonlinear Equation

A Thesis

Submitted to the College of Graduate and Postdoctoral Studies

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in the

Department of Mechanical Engineering

University of Saskatchewan

Saskatoon

by

Jackie Wang

Permission to Use

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor who supervised my thesis work or in his absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying, publication or use of this thesis, or parts thereof, for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Disclaimer

Reference in this thesis to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement, recommendation, or favouring by the University of Saskatchewan. The views and opinions of the author expressed herein do not state or reflect those of the University of Saskatchewan, and shall not be used for advertising or product endorsement purposes.

Requests for permission to copy or to make other uses of materials in this thesis in whole or in part should be addressed to:

Head of the Department of Mechanical Engineering

57 Campus Drive, University of Saskatchewan

Saskatoon, Saskatchewan

Canada, S7N 5A9

OR

Dean

College of Graduate and Postdoctoral Studies

University of Saskatchewan

116 Thorvaldson Building, 110 Science Place

Saskatoon, Saskatchewan

Canada, S7N 5C9

Abstract

Nonlinear equations are known to be difficult to solve, and numerical methods are used to solve systems of nonlinear equations. The objective of this research was to evaluate the potential application of a hybrid mathematical approach to improve the stability of nonlinear equations, with a focus on inverse kinematics. This hybrid approach combines the Newton's method, an existing iterative technique, with the Vector Epsilon Algorithm, a type of convergence accelerator. However, problems arise when the method diverges. In this research, four numerical methods (all based on the classical Newton's method) were used to solve 3 cases studies: 1) a sinusoidal function, which is fundamental to kinematic analyses where position vectors are defined by trigonometric functions; 2) a robot arm with two links pivoted about a pin; and 3) a robot arm with two links on a changeable pole.

For single degree-of-freedom problem, the Newton's method was not able to converge to the closest root when the initial guess was close to a critical point. However, other numerical methods, such as the hybrid method, were able to converge.

Throughout the research, inverse kinematics problems were solved, and results are presented for both existing and new methods.

Acknowledgement

I would like to thank my supervisors Professors A. Dolovich and J.D. Johnston for their patience and guidance. I would also like to acknowledge my advisory committee members, Professors F. Wu, C. Zhang and the external examiner, Professor R.J. Spiteri, for their time. Professor Dolovich for proposing this research and for financial support; the University of Saskatchewan for financial support through a graduate scholarship and the Government of Saskatchewan for financial support. Finally, I would like to thank my family for their support.

Table of Contents

List of Figures

List of Tables

Glossary

ODEs	Ordinary differential equations
PDEs	Partial differential equations
L	Linear operator
$\boldsymbol{x}, \boldsymbol{y}$	Vectors of unknown parameters $x_1, x_2, \ldots, x_n$ and $y_1, y_2, \ldots, y_n$ respectively
$\boldsymbol{g}$	Vector of numbers
a, b, c	Constants
NR	Newton's method
MNR	Modified version of Newton's method
HMNR	J.H. He's modified Newton-Raphson method
Homotopy	Newton-Homotopy method
VEA	Vector Epsilon Algorithm
MNR-VEA	A modified Newton's method combined with the Vector Epsilon Algorithm
CCD	Cyclic coordinates descent algorithm
DOF	Degree of freedom
x_{old}	Value from the previous iteration in NR (1 DOF)
x_{new}	New value of x in the next iteration in NR (1 DOF)
$f(x_{\text{old}})$	Function evaluated at x_{old} in NR (1 DOF)
$f'(x_{\text{old}})$	First derivative evaluated at x_{old} in NR (1 DOF)
$\boldsymbol{x}_{\text{old}}$	Vector from the previous iteration in NR
$\boldsymbol{x}_{\text{new}}$	New vector $\boldsymbol{x}_{\text{in}}$ the next iteration in NR
$\mathbf{F}(\boldsymbol{x}_{\text{old}})$	Function evaluated at $\boldsymbol{x}_{\text{old}}$ in NR

$\mathbf{J}^{-1}$	Inverse of the Jacobian matrix evaluated at $\boldsymbol{x}_{\text{old}}$ in NR
$\mathbf{J}$	Jacobian matrix in NR
$\frac{\partial \mathrm{F}_1}{\partial x_1},\cdots,\frac{\partial \mathrm{F}_n}{\partial x_n}$	Elements of the Jacobian matrix
F_i	Nonlinear functions
x_0	Initial guess
$\hat{x}*$	Solution or root of an equation
f'	First derivative (slope of the tangent) at x_0
$f(x_0)$	Function evaluated at initial guess, x_0
$H(x,t)$	Homotopy function (1 DOF)
$f(x)$	Nonlinear function
$g(x)$	Auxiliary function in Homotopy (1 DOF)
t	Parameter in Homotopy
r	A constant selected by user in HMNR
x_{n+1}^r	Value from the previous HMNR iteration to the power r(1 DOF)
x_n^r	New value of x from HMNR iteration to the power r (1 DOF)
$f(x_n)$	Function evaluated at x_n in HMNR (1 DOF)
$f'(x_n)$	First derivative evaluated at x_n in HMNR (1 DOF)
u	Change in variable $u = x^r$
α	Relaxation parameter in MNR-VEA

SEA	Scalar Epsilon Algorithm
x_n	First order transient
$\tilde{x}$ *	Limit of the sequence x_n
$a\rho^n$	Function of *n* that goes to zero in the limit as *n* to infinity if the absolute value of ρ is less than 1
A, B, C	Iterates from original sequence
D, E	Intermediate values for SEA
F	Predicted limit of the original sequence
A, B, C	Iterates from original sequence in vector form
D, E	Intermediate values for VEA in vector form
F	Predicted limit of the original sequence in vector form
$(\mathbf{E}-\mathbf{D})^+$	Generalized inverse
$\|\mathbf{E}-\mathbf{D}\|$	Euclidean norm
ε	Tolerance
x^C	Critical point
PD	Percentage difference
Δt	Time increment
L_1, L_2	Links in 2, 3 DOF problems
β, γ	Angles in 2, 3 DOF problems
x_P, y_P, z_P	Coordinates of Gripper P in 2, 3 DOF problems
$\boldsymbol{x}$ *	Closest root in 2, 3 DOF problems

$\tilde{\tilde{\boldsymbol{x}}}_i$	Solutions with the domain
$\tilde{\rho}_i$	Distance between $\tilde{\tilde{\boldsymbol{x}}}_i$ and $\boldsymbol{x}^{(0)}$
$\hat{\boldsymbol{x}}$	Any calculated solution from a numerical method
ρ	Distance between $\hat{\boldsymbol{x}}$ and $\boldsymbol{x}^*$
x^{**}	Closest root in 2/1 DOF problem
c	Changeable pole length (3 DOF)

1 Introduction

1.1 Motivation for Research – Challenges in Solving Nonlinear Equations

A combination of measurements and modeling is often used when dealing with engineering problems. When taking measurements, engineers design and set up laboratory equipment to produce data which are collected and analyzed. When modeling, engineers work to understand the theories and principles underlying the physical phenomena, and write or use computer programs to perform virtual experiments as an aid to ensure that results from the physical experiments are reasonable.

Mathematical models for engineering applications, such as in the areas of finite elasticity and inverse kinematics, are often expressed in terms of systems of nonlinear equations which are difficult to solve. There is a problem with uniqueness and existence because a nonlinear system can have multiple solutions or no solution at all. When engineers are conducting numerical research, they either write their own programs or use available computer packages to solve nonlinear systems. Numerical analysis is at the core of both methods, and is directed towards developing and improving the mathematical algorithms required to perform the associated calculations.

The mathematical methods for solving engineering problems can be divided into two main categories: (1) linear and (2) nonlinear systems. The term "systems of equations" might refer to ordinary differential equations (ODEs), partial differential equations (PDEs), integral equations, and/or algebraic equations. In this thesis, the mathematics of nonlinear algebraic equations

arising in engineering applications were investigated. To give a definition for nonlinear algebraic equations, the definition of linear operator must first be reviewed.

Consider a system of equations written as

$$\mathsf{L}(\boldsymbol{x}) = \boldsymbol{g}, \tag{1.1}$$

where L is an operator, $\boldsymbol{x}$ is the vector of unknown parameters $x_1, x_2, \ldots, x_n$, and $\boldsymbol{g}$ is a vector of numbers. The operator L is linear if and only if it satisfies the properties of a linear operator, namely

$$\mathsf{L}(\boldsymbol{x} + \boldsymbol{y}) = \mathsf{L}(\boldsymbol{x}) + \mathsf{L}(\boldsymbol{y}), \tag{1.2}$$

and

$$\mathsf{L}(c\boldsymbol{x}) = c\mathsf{L}(\boldsymbol{x}), \tag{1.3}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors for two separate sets of parameters. If L is a linear operator, then the system of equations given by (1.1) is also linear. In any case where the operator on the left hand side of (1.1) does not satisfy one or both of the properties (1.2) and (1.3), the resulting system is nonlinear. Important examples of nonlinear systems arising in engineering applications include polynomial equations of order greater than one, and equations with trigonometric functions.

Although the mathematics of nonlinear differential and integral equations are still current topics of engineering research, nonlinear algebraic equations still pose a considerable challenge. Only a few analytical solutions exist. As an example, consider a second order polynomial (quadratic equation),

$$ax^2 + bx + c = 0. \tag{1.4}$$

The closed-form expression for x is

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{1.5}$$

which has two solutions. Moreover, real solutions exist if, and only if, $b^2 - 4ac \geq 0$. That is, even for the lowest order polynomial (nonlinear) equation, both uniqueness and existence are issues but not for linear equations (See Section 2.2.1). Analytical solutions also exist for cubic [1] and quartic [2] equations, but according to the Abel-Ruffini theorem [3], no closed-form solution exists for an order higher than four. Similarly, for complicated trigonometric equations (to be described in detail later in the thesis), closed-form solutions are difficult to obtain, or do not exist. Hence, numerical methods are needed, and research into improving numerical methods is needed as well.

1.2 Numerical Methods

Numerical methods, and in particular iterative methods, are used to determine the solution to nonlinear systems arising in engineering applications since there is usually no analytical solution available. Issues with current numerical techniques include the rate of convergence and the uniqueness of the solution. Throughout the centuries, mathematicians and scientists have been using the well-known Newton's method (NR), an iterative technique to determine the solution to nonlinear equations. There exist other numerical schemes to determine the roots of nonlinear systems, such as modified versions of the Newton's method (MNR) and Newton-Homotopy continuation methods, but the NR method is by far the popular choice among academics and industries due to its rapid rate of convergence. The algorithm, however, depends on the initial guess, and neither stability nor convergence is guaranteed.

The iterative methods that were investigated in this research included the Newton's method (NR), J.H. He's modified Newton-Raphson method (HMNR) [4], a Newton-Homotopy continuation method by Wu [5] (to be referred to as "Homotopy" in this thesis), and the new method proposed here – a modified Newton's method combined with the Vector Epsilon Algorithm (MNR-VEA). The engineering application considered in this thesis is the inverse kinematics of robot arms. A review of the literature showed that NR is the method normally used for inverse kinematic calculations (e.g., [6, 7]), and, occasionally, Homotopy is also used (e.g., [8]). This has provided the rationale for the choice of iterative methods considered in this thesis.

1.3 Objective

The research objective was to combine an existing iterative technique, a modified Newton's (MNR) method, with a type of convergence accelerator, the Vector Epsilon Algorithm (VEA), with the aim of improving the stability of nonlinear equations when performing inverse kinematics. The research question is whether the proposed hybrid method (MNR-VEA) is better able to find all the roots of a system of equations when compared to existing methods. This is a pilot study to determine whether MNR-VEA shows any promise at improving stability of nonlinear solutions.

1.4 Scope of Project

The four iterative methods, NR, MNR, Homotopy and the hybrid method MNR-VEA, were tested on three sets of nonlinear equations, which are (i) a single degree of freedom problem represented by a sinusoidal function; (ii) a two degree of freedom problem involving a robot arm

with two links; and (iii) a three degree of freedom problem involving two links with an extendable arm.

The purpose of Case (i) was to assess the performance of each method and the viability of the hybrid method for the fundamental building block of kinematic equations, namely a sinusoidal function. The purpose of Cases (ii) & (iii) were to increase the complexity and further investigate the performance of the algorithms. Both Cases (ii) & (iii) involved position analyses of an end-effector, with the goal of determining all possible configurations that would give the desired coordinates of that end-effector.

1.5 Thesis Style and Organization

Due to the nature of this work, this thesis follows a format often found in the area of applied mathematics and numerical analysis (See references [9, 10] for examples of theses in applied mathematics). This mathematical format is expressed in the style used to present the results and discussion, as well as the manner in which the thesis is organized.

Chapter 2 gives a brief overview of the iterative methods that will be analyzed in the thesis: NR, MNR, Homotopy and MNR-VEA. A brief background is provided for NR, and the mathematical formulation for each technique is presented. In addition, a simple example is given for each method.

The introduction of each iterative method in Chapter 2 is followed by Chapter 3 which gives a detailed description of the test cases within inverse kinematics to be considered; i.e., Cases (i), (ii) & (iii). Chapter 3 also provides the results for each case together with a discussion of performance. Chapter 4 gives a summary and limitations of the findings. Finally, Chapter 5 provides study conclusions and future recommendations.

2 Iterative Methods

2.1 Background and Literature Review

In this section, the techniques for solving nonlinear equations, the Vector Epsilon Algorithm (VEA) and some inverse kinematics applications are discussed. Notation for system of nonlinear equations in here is $\mathbf{F}(\boldsymbol{x}) = 0$. Numerous numerical methods are based on the Newton's method, and some of them will be reviewed in upcoming sections. The book by Kelley [11], S*olving nonlinear equations with Newton's method*, is an introductory textbook for academics who are working on numerical analysis. Some of the algorithms are written in pseudocode and MATLAB® [12] code for users to experiment with. A Newton iteration [11] needs several steps:

- evaluate the function at each iteration, $\mathbf{F}(\boldsymbol{x}_n)$, and a test for termination;
- approximate solution of the equation

$$\mathbf{F}'(\boldsymbol{x}_n)\boldsymbol{s}_n = -\mathbf{F}(\boldsymbol{x}_n) \tag{2.1}$$

  where $\boldsymbol{s}_n$ is Newton step; and
- obtain $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \lambda \boldsymbol{s}_n$, where the step length λ is a positive scalar to ensure a decrease in $\|\mathbf{F}\|$.

The calculation of Newton step can be expensive sometimes, and step length is not needed if Newton step is reasonable. The norm of $\mathbf{F}(\boldsymbol{x})$is an indicator of the rate of decay in error [13]. The termination criterion used in [11] is

$$\|\mathbf{F}(\boldsymbol{x}_n)\| \le \tau_r \|\mathbf{F}(\boldsymbol{x}_0)\| + \tau_a, \tag{2.2}$$

where the relative τ_r and absolute τ_a error tolerances chosen by user.

2.1.1 Solving Scalar Equation

For a scalar equation $f(x) = 0$, the Newton's method is represented by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \tag{2.3}$$

where x_n is the value from the previous iteration, $f(x_n)$ is the function evaluated at x_n, $f'(x_n)$ is the first derivative evaluated at x_n, and x_{n+1} is the new value of *x*. The damped Newton's method [14] has a damping factor λ, also known as step length,

$$x_{n+1} = x_n - \lambda \frac{f(x_n)}{f'(x_n)}, \tag{2.4}$$

to control the Newton step from going too far. The method evaluates

$$\|\mathbf{F}(x_n + \lambda_m s_n)\| < (1 - \alpha \lambda_m)\|\mathbf{F}(x_n)\| \tag{2.5}$$

where the parameter $\alpha \in (0,1)$ is small enough to satisfy condition (**2.5**) and λ_m is the minimum of $\phi(\lambda) = \|\mathbf{F}(x_n + \lambda s_n)\|^2$ [13]. There are methods that solve scalar equation without calculating the derivative, such as (i) bisection or binary-search method [15],

$$x = \frac{a-b}{2}, \tag{2.6}$$

where f is continuous on $[a,b]$ and points a, b, x update after each iteration; (ii) secant method [15] uses a finite difference to estimate $f'(x_n)$ that

$$b_n = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}, \tag{2.7}$$

and the new iterate is

$$x_{n+1} = x_n - b_n^{-1} f(x_n); \tag{2.8}$$

and (iii) inverse quadratic interpolation method [16] is

$$\begin{aligned} x_{n+1} = & \frac{f(x_{n-1}) f(x_n)}{(f(x_{n-2}) - f(x_{n-1}))(f(x_{n-2}) - f(x_n))} x_{n-2} \\ & + \frac{f(x_{n-2}) f(x_n)}{(f(x_{n-1}) - f(x_{n-2}))(f(x_{n-1}) - f(x_n))} x_{n-1}, \\ & + \frac{f(x_{n-2}) f(x_{n-1})}{(f(x_n) - f(x_{n-2}))(f(x_n) - f(x_{n-1}))} x_n \end{aligned} \tag{2.9}$$

which uses quadratic interpolation (requires three points) to find the root. Scalar equations can be solved in MATLAB by using *fzero* which is based on Brent's method [17], a hybrid root-finding algorithm that combines methods (i) - (iii).

2.1.2 Solving System of Nonlinear Equations

For a system of nonlinear equations, all quantities are written in terms of vectors and matrices, and the Newton's method is

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \mathbf{J}^{-1}(\boldsymbol{x}_n)\, \mathbf{F}(\boldsymbol{x}_n), \tag{2.10}$$

where $\boldsymbol{x}_n$, $\boldsymbol{x}_{n+1}$, $\mathbf{F}(\boldsymbol{x}_n)$ have the same definitions as for equation (2.3), but expressed in vector form, and $\mathbf{J}^{-1}$ is the inverse of the Jacobian matrix evaluated at $\boldsymbol{x}_n$. The damped Newton's method is equation (2.4) in vector form

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \lambda\, \mathbf{J}^{-1}(\boldsymbol{x}_n)\, \mathbf{F}(\boldsymbol{x}_n). \tag{2.11}$$

Broyden's method [18] is a generalization of the secant method in higher dimensions, expressed as equation (2.7) in vector form,

$$\mathbf{B}_n\left(\boldsymbol{x}_n-\boldsymbol{x}_{n-1}\right)=\mathbf{F}\left(\boldsymbol{x}_n\right)-\mathbf{F}\left(\boldsymbol{x}_{n-1}\right), \tag{2.12}$$

and it is a quasi-Newton algorithm [15] where the Jacobian is replaced with an approximation matrix

$$\boldsymbol{x}_{n+1}=\boldsymbol{x}_n-\mathbf{B}_n^{-1}\mathbf{F}\left(\boldsymbol{x}_n\right). \tag{2.13}$$

After obtaining $\boldsymbol{x}_{n+1}$, $\mathbf{B}_{n+1}$ can be calculated using

$$\mathbf{B}_{n+1}=\mathbf{B}_n+\frac{\left(\boldsymbol{y}-\mathbf{B}_n\boldsymbol{s}\right)\boldsymbol{s}^{\mathrm{T}}}{\boldsymbol{s}^{\mathrm{T}}\boldsymbol{s}}, \tag{2.14}$$

where $\boldsymbol{y}=\mathbf{F}\left(\boldsymbol{x}_{n+1}\right)-\mathbf{F}\left(\boldsymbol{x}\right)$ and $\boldsymbol{s}=\boldsymbol{x}_{n+1}-\boldsymbol{x}_n$. An inexact Newton method [19] by Dembo et al. is an extension of Newton's method. A parameter called a forcing term, η_n, is introduced to control efficiency and robustness. The method needs to satisfy the following condition

$$\left\|\mathbf{F}\left(\boldsymbol{x}_n\right)+\mathbf{F}'\left(\boldsymbol{x}_n\right)\boldsymbol{s}_n\right\|\le\eta_n\left\|\mathbf{F}\left(\boldsymbol{x}_n\right)\right\|, \tag{2.15}$$

where $\eta_n\in[0,1)$. Inexact Newton method is the Newton's method if $\eta_n=0$ for all n. Eisenstat and Walker [20] came up with 2 choices for η_n. The first alternative

$$\eta_n=\frac{\left\|\mathbf{F}\left(\boldsymbol{x}_n\right)-\mathbf{F}\left(\boldsymbol{x}_{n-1}\right)-\mathbf{F}'\left(\boldsymbol{x}_{n-1}\right)\boldsymbol{s}_{n-1}\right\|}{\left\|\mathbf{F}\left(\boldsymbol{x}_{n-1}\right)\right\|}, \tag{2.16}$$

and

$$\eta_n=\frac{\left|\left\|\mathbf{F}\left(\boldsymbol{x}_n\right)\right\|-\left\|\mathbf{F}\left(\boldsymbol{x}_{n-1}\right)+\mathbf{F}'\left(\boldsymbol{x}_{n-1}\right)\boldsymbol{s}_{n-1}\right\|\right|}{\left\|\mathbf{F}\left(\boldsymbol{x}_{n-1}\right)\right\|} \tag{2.17}$$

reflect the agreement between $\mathbf{F}(\boldsymbol{x})$ and its local linear model at the previous step. The other alternative

$$\eta_n = \gamma \left(\frac{\|\mathbf{F}(\boldsymbol{x}_n)\|}{\|\mathbf{F}(\boldsymbol{x}_{n-1})\|} \right)^{\alpha} \tag{2.18}$$

where $\gamma \in [0,1]$ and $\alpha \in (1,2]$ control the rate of decrease of $\{\eta_n\}$. Small forcing terms cause unnecessary computation and might lead to failure. On the other hand, large forcing terms may lessen the number of computation but increase the number of $\boldsymbol{s}_n$ for convergence.

A system of nonlinear equations can be solved in MATLAB by *fsolve*. It implements the trust region dogleg and Levenberg-Marquardt methods. The trust region dogleg method [21-23] is

$$\min \left[\frac{1}{2}\mathbf{F}^{\mathrm{T}}\mathbf{F} + \boldsymbol{s}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}\mathbf{F} + \frac{1}{2}\boldsymbol{s}^{\mathrm{T}}\mathbf{J}^{\mathrm{T}}\mathbf{J}\,\boldsymbol{s} \right] \text{subject to } \|\mathbf{D}\,\boldsymbol{s}\| \le \Delta, \tag{2.19}$$

where $\mathbf{D}$ is a diagonal scaling matrix and Δ is the trust region bound. Levenberg-Marquardt method [24, 25], also known as the damped least squares method, combines the gradient descent and Gauss-Newton methods,

$$\left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda \mathbf{I}\right)\boldsymbol{s} = -\mathbf{J}^{\mathrm{T}}\mathbf{F} \tag{2.20}$$

where λ is the damping parameter $(\lambda \ge 0)$, $\mathbf{I}$ is the identity matrix, and $\boldsymbol{s}$ is the solution to

$$\min \frac{1}{2}\|\mathbf{J}\,\boldsymbol{s} + \boldsymbol{F}\|^2 \text{ subject to } \|\boldsymbol{s}\| \le \Delta. \tag{2.21}$$

2.1.3 Applications with Vector Epsilon Algorithm

The Vector Epsilon Algorithm (VEA) is an efficient method in accelerating the convergence of vector sequences [26]. Later on, Gekeler [27] showed that the VEA was able to accelerate

convergence when solving non-singular systems of linear and non-linear equations. As for using the VEA to accelerate convergence when solving singular systems, Brezinski [28] and Sidi [29] showed success in systems of linear equations, and Brezinski [30] for systems of non-linear equations. Brezinski and Redivo-Zaglia [31] released a MATLAB toolbox named EPSfun that included codes for Scalar Epsilon Algorithm (SEA) and VEA. Waldvogel [32] used the VEA for exploration of data to lessen the computation time with their own MATLAB code. The algorithm has been applied to engineering applications, such as in fluid dynamics applications, with Hafez and Cheng [33] and Hafez et al. [34] using the SEA to reduce solving time in transonic flow calculations; Cheung [35] used the VEA to reduce solving time in viscous and inviscid hypersonic flow calculations. The algorithm has also been applied to kinematic problems, which is the focus of this thesis.

2.1.4 Inverse Kinematics Problems

Kinematics describes the motion of bodies within a system without consideration of the forces causing the motion. Hence, kinematics is the study of motion based on geometry and changes in geometry. The motion of each body is modelled through mathematical formulas for calculating position, velocity, and acceleration. The area of kinematics can be divided into forward and inverse problems. When positions are the primary consideration for a mechanism, forward kinematics is a straightforward process: given a set of joint angles and link parameters defining a configuration, the aim is to find the position of an end-effector. The inverse problem is the reverse of the forward kinematic process: given the end-effector position, find the joint angles and link parameters to achieve that position [36]. In this thesis, the focus will be on inverse

kinematics which, as will be shown, is highly nonlinear and therefore is the more challenging problem.

In the literature, there are several techniques used to perform inverse kinematics. A problem of particular interest is the inverse kinematics of robot manipulators, and different iterative methods have been used to solve the corresponding nonlinear equations. Cai et al. [37] and Lenarcic [38] solved nonlinear kinematic equations using iterative algorithm based on the conjugate gradient method. This method [39] solves systems of linear equations in the form of $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ where $\mathbf{A}$ is a symmetric and positive-definite matrix. It is equivalent to finding the minimum of quadratic form

$$\mathbf{F}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\mathbf{A}\,\boldsymbol{x} - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{b}, \tag{2.22}$$

$$\mathbf{F}'(\boldsymbol{x}) = \mathbf{A}\,\boldsymbol{x} - \boldsymbol{b}. \tag{2.23}$$

The minimum of $\mathbf{F}(\boldsymbol{x})$ is a solution to $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$. Caccavale and Wroblewski [40] compared the effectiveness and robustness of Newton-Raphson and Jacobian transpose methods, where they used the transpose instead of the inverse of a Jacobian [41], for determining the roots of nonlinear kinematic equations. Tagawa and Haneda [42] developed a computer program to implement a fast interval bisection (FIB) algorithm based on interval analysis. It reduced the number of operations and storage space of variables. Cai and Zhang [43] programmed a solver based on the gradient descent method in neural networks, and Martin el al. [44] used the gradient descent method to solve the inverse kinematics of multi-link robots by means of neuro-controllers. Chu and Wu [45] showed that a modified secant method has a better performance, assessed via several numerical examples, than the Newton's method. Ren et al. [46] used the cyclic coordinates descent (CCD) algorithm to perform inverse kinematics for virtual human

running. Olsen et al. [47] used both numerical and analytical CCD to determine the required angles at robot joints.

In this thesis, a new method to solve inverse kinematics equations will be introduced, and the main objective was to test the feasibility of the new method. In order to do this, the new method was compared to fundamental iterative methods. Four types of iterative schemes were therefore studied:

1. Newton's method (NR);
2. Newton-Homotopy method (Homotopy);
3. He's Modified Newton-Raphson method (HMNR); and
4. The New Proposed Method – Modified Newton's method with Vector Epsilon Algorithm (MNR-VEA).

NR is commonly used when an iterative method is needed to solve certain problems. The reason for its popularity is due to its fast convergence. Wu's version of Homotopy is similar to NR in some ways, in that it provides better performance since it is independent of initial guesses and always converges. HMNR is an altered form of NR, and it operates like NR except there is a parameter which controls the iterative process. The proposed method in this thesis, MNR-VEA, combines a modified version of Newton's method with a type of convergence accelerator, the Vector Epsilon Algorithm. Numerical analysis was performed for the investigation of the above iterative techniques. One of the primary issues that will form the basis for comparison is the need for finding *all* the roots of a system of nonlinear equations. For example, for the application of a robot manipulator, this would correspond to finding *all* the configurations that would achieve a specified position of the end effector. The "best" configuration could then be chosen based on some optimization criterion. As will be seen, this is a particular challenge for

any iterative method, and depends on the technique's ability to find the closest root to a given initial guess.

2.2 Solutions of Linear and Nonlinear equations

2.2.1 Existence and Uniqueness of Solutions

Fundamental theorem for linear systems [48] says a general system of linear equations with *n* unknowns can be written as

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \ , \\ &\vdots \\ \boldsymbol{a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n} &= \boldsymbol{b_n} \end{aligned} \tag{2.24}$$

where $a_{11}, a_{12}, \ldots, a_{nn}$ are the coefficients, $x_1, x_2, \ldots, x_n$ are the unknowns, $b_1, b_2, \ldots, b_n$ are the constants. The system is consistent if the coefficient matrix $\mathbf{A}$ and the augmented matrix $\hat{\mathbf{A}}$ have the same rank where

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \text{ and } \hat{\mathbf{A}} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{bmatrix}.$$

The system has a unique solution if and only if the rank of matrices $\mathbf{A}$ and $\hat{\mathbf{A}}$ is *n*. The system has infinitely many solutions if the rank of matrices $\mathbf{A}$ and $\hat{\mathbf{A}}$ is less than *n*. The system can be solved using Gaussian elimination [49].

The existence and uniqueness of the solutions of nonlinear equations can be explained by the contraction mapping theorem [50], also known as the fixed point theorem. Let $\boldsymbol{I}$ be a closed

interval and let f be a mapping of $\boldsymbol{I}$ into itself. If there exists a real number $\lambda < 1$such that for all $x, y \in \boldsymbol{I}$, the Lipschitz condition

$$\left| f(x) - f(y) \right| \le \lambda \left| x - y \right|, \tag{2.25}$$

is satisfied, where λ is the Lipschitz constant and $\lambda < 1$. Then

1. f is said to be a contraction mapping;
2. f has one and only one fixed point, i.e., there exists a unique solution $x^* \in \boldsymbol{I}$ such that $f(x^*) = x^*$;
3. any sequence given iteratively by $x^{(k+1)} = f\left(x^{(k)}\right)$ where $x^{(0)} \in \boldsymbol{I}$, converges to x^* as $k \to \infty$;
4. a priori bound for the distance between $x^{(k)}$ and x^* is given by

$$\left| x^{(k)} - x^* \right| \le \frac{\lambda^k}{1-\lambda} \left| x^{(1)} - x^{(0)} \right| \tag{2.26}$$

and a posteriori bound is given by

$$\left| x^{(k)} - x^* \right| \le \frac{\lambda}{1-\lambda} \left| x^{(k)} - x^{(k-1)} \right|. \tag{2.27}$$

Extended to systems of nonlinear equations $\mathbf{F}(\boldsymbol{x}) = 0$, and rewrite to $\boldsymbol{x} = \mathbf{F}(\boldsymbol{x})$, i.e.,

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \mathrm{F}_1(x_1, x_2, \cdots, x_n) \\ \mathrm{F}_2(x_1, x_2, \cdots, x_n) \\ \vdots \\ \mathrm{F}_n(x_1, x_2, \cdots, x_n) \end{bmatrix}, \tag{2.28}$$

the Lipschitz condition now becomes

$$\left\| \mathbf{F}(\boldsymbol{x}) - \mathbf{F}(\hat{\boldsymbol{x}}) \right\| \le \lambda \left\| \boldsymbol{x}_i - \hat{\boldsymbol{x}}_i \right\|, \tag{2.29}$$

where $\lambda < 1$ and the norm $\|\boldsymbol{x_i} - \hat{\boldsymbol{x}}_{\boldsymbol{i}}\| = \max_i |x_i - \hat{x}_i|$, it is called the maximum norm and can be written as $\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_\infty$. The Lipschitz condition in the above can be replaced with

$$\max_i \sum_{j=1}^{n} \left| \frac{\partial \mathrm{F}_i}{\partial x_i} \right| \leq \lambda < 1. \tag{2.30}$$

If the sum of the derivative is less than 1, then there exists a unique solution $\mathbf{F}(\boldsymbol{x}^*) = \boldsymbol{x}^*$ if it maps to itself.

2.2.2 The Issue of Closest Root

Numerical methods are used to solve equations involving trigonometry functions, such as load flow analysis [51, 52] and inverse kinematics [53, 54]. There is a need to find the closest solution from the given initial guess of nonlinear equations. For Chemical Engineering applications, finding the closest solution is essential in the synthesis, design and control of chemical processes [55, 56] and useful roots can be selected [57]. For Electrical Engineering applications, finding the closest solution is crucial to computer-aided design of integrated circuits since they are the operating points [58-61].

In order to find solutions within a certain domain systematically, a common approach is to form a grid of points which are used as a series of initial guesses. For each initial guess, the goal is to use the iterative method to find the root closest to that initial guess. If the iterative method does not give the closest root relative to the initial guess, the process is no longer systematic, and it cannot be concluded that all the roots have been found. For example, the classical numerical method NR is dictated by the initial guess used in the iterative process. This method fails when a singularity occurs and is highly unstable near a singularity. It could result in a root distant from

the initial guess that may even be outside the defined domain. Due to this, in the literature there are numerous versions of MNR which try to eliminate unfavourable qualities from the original recipe.

To illustrate the meaning of closest root, consider Figure 2.1 and Figure 2.2. Both figures are illustrating the solution of a system

$$\begin{aligned} f(x,y) &= 0 \\ g(x,y) &= 0 \end{aligned} \tag{2.31}$$

of two nonlinear equations in two unknowns. A two degree of freedom (2-DOF) system is considered for this illustration, since it is easier to visualize the numerical approach for two variables as compared to the difficulty in drawing a function of three or more variables. Figure 2.1 demonstrates that, depending on the iterative method, an initial guess may not give the closest root, and in fact the root finding process may become random. Figure 2.2 demonstrates a case where all initial guesses are able to converge to the closest root. Here, by definition, the closest root occurs when the Euclidean distance between the initial guess and the solution is the smallest. In the sections to follow, each iterative method to be considered in this thesis is presented and, later in the thesis, each method was evaluated in terms of its ability to find the closest root for an initial guess.

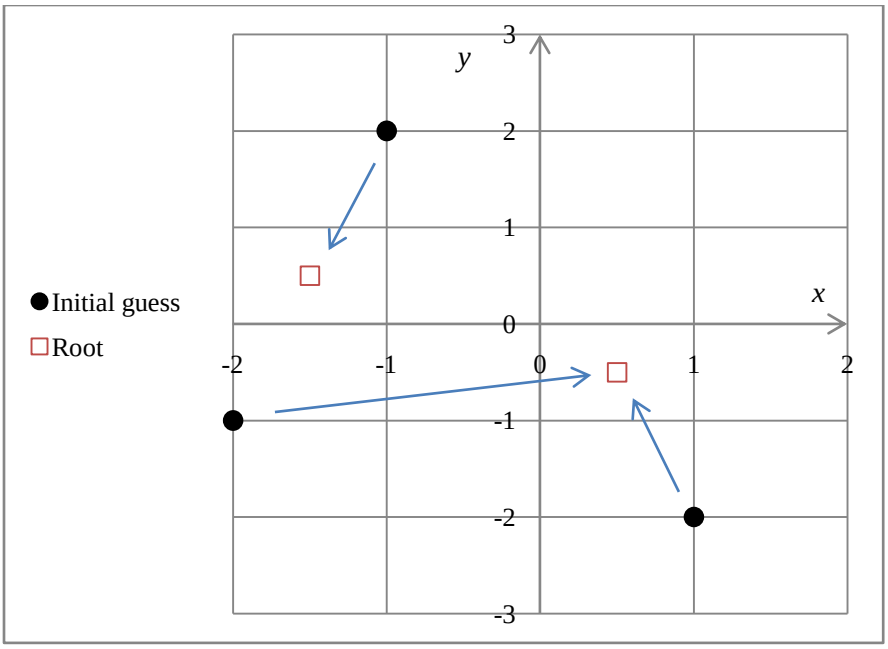

Figure 2.1: A case where initial guesses do not all converge to the closest root.

All roots in the domain are shown in the figure.

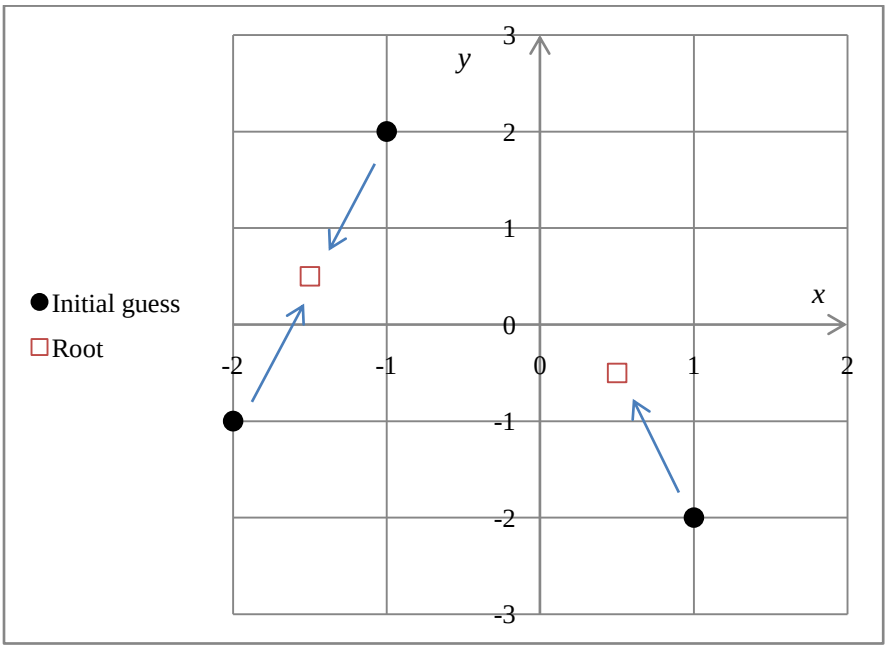

Figure 2.2: A case where all initial guesses converge to the closest root.

All roots in the domain are shown in the figure.

2.3 Newton's Method

The classical Newton's method is also known as the Newton-Raphson method for solving a single nonlinear equation, named after Isaac Newton and Joseph Raphson [62]. Going back to the year 1600, François Viète came up with a perturbation technique for solving scalar polynomial equations. Newton learnt Vieta's method in 1664, and by 1669 had improved the method to achieve quadratic convergence [63]. Raphson transformed the method into an improved iterative scheme. Up until then, the concept of derivative had not been incorporated into the technique. Thomas Simpson implemented the method with calculus in 1740 and it has been widely used ever since. Equations (2.3) and (2.10) are Newton's method for solving single and system of nonlinear equations.

The role of the inverse of the Jacobian matrix for a nonlinear system is analogous to that of the first derivative for a single nonlinear equation. The general mathematical expression for the Jacobian matrix is

$$\mathbf{J} = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial x_1} & \cdots & \frac{\partial F_n}{\partial x_n} \end{bmatrix}, \tag{2.32}$$

where the elements of the matrix are partial derivatives. Since this research is a pilot study and involves at most three variables, the Jacobian matrix is a 3×3 matrix and each function, F_i, has variables x_1, x_2, x_3; that is

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathrm{F}_1}{\partial x_1} & \frac{\partial \mathrm{F}_1}{\partial x_2} & \frac{\partial \mathrm{F}_1}{\partial x_3} \\ \frac{\partial \mathrm{F}_2}{\partial x_1} & \frac{\partial \mathrm{F}_2}{\partial x_2} & \frac{\partial \mathrm{F}_2}{\partial x_3} \\ \frac{\partial \mathrm{F}_3}{\partial x_1} & \frac{\partial \mathrm{F}_3}{\partial x_2} & \frac{\partial \mathrm{F}_3}{\partial x_3} \end{bmatrix}. \tag{2.33}$$

Figure 2.3 is a graphical representation of a single equation $f(x) = 0$, where x^* is the solution (i.e., root) of the equation. Initial guess x_0 is located on the x-axis, and the value of f and its derivative f' (slope of the tangent) at x_0 are calculated. The x-intercept of the tangent line at point $(x_0, f(x_0))$ is calculated as the next iterate for the method. The process is repeated until the method converges.

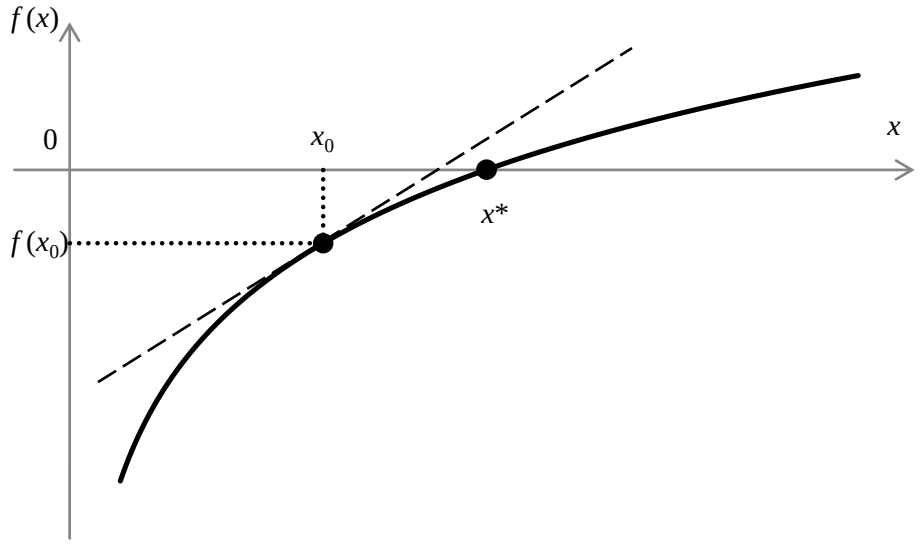

Figure 2.3: An illustration of the Newton's method. The algorithm begins with initial guess x_0, and the next iterate is calculated using the value at x_0 and the slope at x_0. This process continues until the method reaches solution x^*.[64]

2.4 Newton-Homotopy Continuation Method

Another iterative process is named the Newton-Homotopy Continuation method (Homotopy) [5]. It is a technique used to eliminate the possibility of bad initial guesses which could prevent achieving the closest root. The method begins with a simple auxiliary function $g(x)$ where the answer to auxiliary equation $g(x)=0$ can easily be found. In the next step, the answer to $g(x)=0$ is used as the initial guess for finding the roots of a new intermediate function which is closer, by some criterion, to the function of interest, $f(x)$. The method continues, incrementally approaching $f(x)$ with a series of intermediate functions. At the end of the process, the solution of the last intermediate equation is used as the initial guess for finding the roots of $f(x)$.

Mathematically, the intermediate function is given by

$$H(x,t)=t\ f(x)+(1-t)\ g(x), \tag{2.34}$$

where *x* is the unknown variable which is to be determined iteratively, $f(x)=0$ is the equation to be solved, and $g(x)$ is the auxiliary function. The parameter *t* is introduced to achieve the series of intermediate functions *H*(*x*,*t*). It begins with a value $t=0$ and is updated incrementally ending with a value of $t=1$, i.e.,

$$H(x,0)=g(x), \tag{2.35}$$

$$H(x,1)=f(x). \tag{2.36}$$

Homotopy finds a root for each intermediate function defined through small increments Δ*t*, and uses the root calculated for the previous function as the initial guess in the next step. For each value of *t*, the Newton's method is used to solve *H*(*x*,*t*) = 0. When *t* reaches one, the method is

essentially the NR method applied to $f(x)=0$, but with an initial guess that was obtained through the Homotopy progression of functions.

2.5 He's Modified Newton-Raphson Method

There are numerous versions of modified Newton's methods available in the literature [65], and each has their own uniqueness. The one considered here is He's [4] version of the iterative technique which is given by

$$x_{n+1}^{r} = x_n^{r} - \frac{r\, x_n^{r-1} f(x_n)}{f'(x_n)}, \tag{2.37}$$

where again the equation to be solved is $f(x)=0$, $f(x_n)$is the function evaluated at iterate x_n, $f'(x_n)$ is the first derivative evaluated at x_n, and r is a constant selected by the user. He varied the value of r, and compared the results to Newton's method. He stated the optimal choice of r, proof of convergence and accuracy of the method needed to work out as future work; however, the preliminary study showed potential the find the solution without calculating the Jacobian matrix. In the case of $r=1$, He's method is the original Newton's technique. He implements r as a power and also as a coefficient in the formula. He used different values of r in equation (2.37) and achieved different results. In the numerical tests to be shown later in the thesis, the effect of r was demonstrated in relation to the performance of the new proposed method which is introduced in the next section.

Another interpretation of HMNR is as follows. Consider a change in variable where $u = x^r$ and therefore $x = u^{1/r}$. Then

$$x^{r-1} = x^r\ x^{-1} = \frac{x^r}{x} = \frac{u}{u^{1/r}} = u^{1-\frac{1}{r}} = u^{\frac{r-1}{r}}, \tag{2.38}$$

and

$$f(x) = f(u^{1/r}) = h(u). \tag{2.39}$$

The first derivative of $f(x)$ is

$$f'(x) = \frac{\mathrm{d}f}{\mathrm{d}x} = \frac{\mathrm{d}h}{\mathrm{d}u}\frac{\mathrm{d}u}{\mathrm{d}x} = \frac{\mathrm{d}h}{\mathrm{d}u}\left(r\, x^{r-1}\right). \tag{2.40}$$

Using equations (2.38) and (2.40), therefore, the HMNR as given by equation (2.37) becomes

$$u = u - \frac{r\, u^{\frac{r-1}{r}}\, h(u)}{r\, u^{\frac{r-1}{r}}\, h'(u)}, \tag{2.41}$$

$$u = u - \frac{h(u)}{h'(u)}. \tag{2.42}$$

That is, equation (2.42) shows that HMNR can be interpreted as Newton's method with a change in variable $u = x^r$.

2.6 The Proposed Method: Modified Newton's Method with Vector Epsilon Algorithm

Although the classical Newton's method provides rapid (quadratic) convergence, a common problem is overshooting the closest root. The new method combines a slowly converging Modified Newton's method (MNR), with a convergence accelerator, the Vector Epsilon Algorithm (VEA). The Modified Newton's method approaches the root gradually, and therefore eliminates the chance of overshooting. This technique is slow, and, hence, VEA is introduced to speed up the process. After a few iterates are obtained from MNR, the convergence accelerator extrapolates to the closest root. The MNR and VEA techniques are described in the sections that follow.

2.6.1 A Modified Version of the Newton's Method

The Modified Newton's Method used in the proposed hybrid technique is different from He's method in Section 2.5, as known as damped Newton's method [14]. For single nonlinear equation with one unknown variable, the MNR in the proposed technique is given by

$$x_{\text{new}} = x_{\text{old}} - \alpha \frac{f(x_{\text{old}})}{f'(x_{\text{old}})}, \qquad 0 < \alpha < 1 \tag{2.43}$$

where α is the relaxation parameter, x_{old} is the value from the previous iteration, $f(x_{\text{old}})$ is the function evaluated at x_{old}, $f'(x_{\text{old}})$ is the first derivative evaluated at x_{old}, and x_{new} is the new value of x. This method will approach a solution slowly as the relaxation parameter, α, becomes smaller.

For a system of nonlinear equations, the MNR is given by

$$\boldsymbol{x}_{\text{new}} = \boldsymbol{x}_{\text{old}} - \alpha \mathbf{J}^{-1} \boldsymbol{F}(\boldsymbol{x}_{\text{old}}), \qquad 0 < \alpha < 1. \tag{2.44}$$

where $\boldsymbol{x}_{\text{old}}$, $\boldsymbol{x}_{\text{new}}$, $\boldsymbol{F}$ ($\boldsymbol{x}_{\text{old}}$) have the same definitions as for equation (2.43), but expressed in vector form, and $\mathbf{J}^{-1}$ is the inverse of the Jacobian matrix evaluated at $\boldsymbol{x}_{\text{old}}$.

The issue of picking a specific value for the relaxation parameter, α, will be addressed as part of the results and discussion sections of the thesis. In general, a value between 0 and 1 is selected to ensure relaxation is achieved.

2.6.2 The Vector Epsilon Algorithm

Convergence accelerators are used in various fields [66]. The Shanks Transform was developed in 1955, and then the Scalar Epsilon Algorithm (SEA) and the Vector Epsilon Algorithm (VEA) were developed by Wynn in 1956 and 1962, respectively [26, 67]. VEA is a vector extension of

SEA, and SEA is an algorithmic implementation of Shanks Transform. The properties of SEA have been established; however, the nature of VEA is still a topic of ongoing research. To understand the VEA, the background of Shanks Transform and the SEA must first be reviewed.

Shanks Transform was developed by Daniel Shanks [68] in 1955, but the same algorithm had been published by Schmidt [69] in 1941. It is based on the concept of mathematical transient. A first order transient is written as

$$x_n = x^* + a\rho^n, \tag{2.45}$$

where the term $a\rho^n$ is a function of n that goes to zero in the limit as n to infinity if the absolute value of ρ is less than 1. In this case, x^* is the limit of the sequence x_n, $n = 0, 1, 2 \ldots$

By definition, the first order Shanks Transform extrapolates an iterative process to an approximation of its limit x^* by fitting three iterates to the assumed form given by equation (2.45). That is, three iterates x_1, x_2, and x_3 are written as

$$x_1 = x^* + a\rho^1, \tag{2.46}$$

$$x_2 = x^* + a\rho^2, \tag{2.47}$$

$$x_3 = x^* + a\rho^3. \tag{2.48}$$

The process of determining a formula for x^* begins by calculating the difference between x_1 and x_2 from equations (2.46) and (2.47) giving

$$x_2 - x_1 = a\rho^2 - a\rho^1 \tag{2.49}$$

or

$$x_2 - x_1 = a\rho(\rho - 1) \quad . \tag{2.50}$$

Rearranging equation (2.50) gives

$$a\rho = \frac{x_2 - x_1}{\rho - 1} \tag{2.51}$$

or

$$a\rho = \frac{\Delta x_1}{\rho - 1} \quad , \tag{2.52}$$

where $\Delta x_1 = x_2 - x_1$. Similarly, for x_2 and x_3,

$$x_3 - x_2 = a\rho^3 - a\rho^2 \tag{2.53}$$

$$x_3 - x_2 = a\rho^2(\rho - 1) \tag{2.54}$$

$$a\rho^2 = \frac{x_3 - x_2}{\rho - 1} \tag{2.55}$$

and

$$a\rho^2 = \frac{\Delta x_2}{\rho - 1} \quad . \tag{2.56}$$

Substituting equations (2.52) and (2.56) into equations (2.46) and (2.47) gives

$$x_1 = x^* + \Delta x_1 \frac{1}{\rho - 1} \tag{2.57}$$

$$x_2 = x^* + \Delta x_2 \frac{1}{\rho - 1} \tag{2.58}$$

or

$$x^* + \Delta x_1 \frac{1}{\rho - 1} = x_1 \tag{2.59}$$

$$x^* + \Delta x_2 \frac{1}{\rho - 1} = x_2 \quad . \tag{2.60}$$

Equations (2.59) and (2.60) are two equations with two unknowns. If x^* and $\frac{1}{\rho - 1}$ are treated as the unknowns, then the system is linear, and solution by Cramer's rule gives a formula for the limit, x^*; that is,

$$x^* = \frac{\begin{vmatrix} x_1 & \Delta x_1 \\ x_2 & \Delta x_2 \end{vmatrix}}{\begin{vmatrix} 1 & \Delta x_1 \\ 1 & \Delta x_2 \end{vmatrix}} \quad . \tag{2.61}$$

In general, the limit for any three consecutive iterates, x_m, x_{m+1}, and x_{m+2}, is given by

$$x^* = \frac{\begin{vmatrix} x_m & \Delta x_m \\ x_{m+1} & \Delta x_{m+1} \end{vmatrix}}{\begin{vmatrix} 1 & \Delta x_m \\ 1 & \Delta x_{m+1} \end{vmatrix}} \quad . \tag{2.62}$$

Note that the original system given by equations (2.46), (2.47) and (2.48) was nonlinear in terms of unknowns, a, ρ, and x^*. The process between equations (2.49) and (2.58) transforms the original system into a linear system of equations in terms of x^* and $\frac{1}{\rho - 1}$. Equation (2.62) is one version of Shanks Transform.

An alternate way developed to calculate Shanks Transform without determinants is the Scalar Epsilon Algorithm (SEA). The algorithm is implemented using an array of numbers (scalars) as shown in Table 2.1.

Table 2.1: SEA Transformation Table.

Col. 1	Col. 2	Col. 3	Col. 4
0			
	A		
0		D	
	B		F
0		E	
	C		
0			

The table consists of four columns. The first column is a column of zeroes, and the second column contains three iterates from the original sequence/process, denoted A, B, and C. The third column contains intermediate values D and E which, by definition of the algorithm, are given by

$$D = 0 + \frac{1}{B - A} \tag{2.63}$$

and

$$E = 0 + \frac{1}{C - B} \quad . \tag{2.64}$$

Finally, the fourth column gives the predicted limit F of the original sequence, according to the formula

$$F = B + \frac{1}{E - D} \quad . \tag{2.65}$$

As shown in Appendix A, the approximation given by equation (2.65) for SEA is identical to that given by equation (2.62) for Shanks Transform. Also, it should be noted that the approximation given by equations (2.62) and (2.65) is defined as “first order”, since it is obtained by fitting a

first order transient (equation (2.45)) to the iterates. For higher orders, the assumed form is a k^{th} order transient of the form

$$x_n = x^* + \sum_{i=1}^{k} a_i \rho_i^n \,, \tag{2.66}$$

where k is the order of the transform, and x^*, a_i and ρ_i are to be determined. For a k^{th} order transform, therefore, there are $2k+1$ unknowns in the assumed form, and $2k+1$ iterates are needed to calculate the approximation to the limit x^*. The algorithm for higher orders is similar to that given in Table 2.1 except that more rows are used to accommodate the additional iterates. Calculations are performed using equations such as (2.63) to (2.65) until a single value is achieved in the last column. Higher order SEA may give a better approximation to the limit, depending on the chosen sequence. In addition, it may be shown that the first order transform gives the limit exactly if the iterates are the partial sums of a geometric series. Similarly, a k^{th} order transform gives the limit exactly if the iterates are the partial sums of k geometric series added together. Unfortunately, SEA can only deal with scalar sequences, such as those arising from the iterative solution of a *single* nonlinear equation. For systems of nonlinear equations generating vector sequences, the Vector Epsilon Algorithm (VEA) was developed. Table 2.2 is used to implement VEA, which was also developed by Wynn.

Table 2.2: VEA Transformation Table.

Col. 1	Col. 2	Col. 3	Col. 4
0			
	A		
0		**D**	
	B		**F**
0		**E**	
	C		
0			

Note that in Table 2.2 the quantities are in vector form. The VEA follows the same basic format that was used with Table 2.1 to calculate the transformed quantity **F**. That is, VEA is essentially a vector extension of SEA. However, in defining the VEA, Wynn needed to replace the scalar equations (2.63) to (2.65) with operations that can be applied to vectors. To clarify this, equation (2.65) can be written in another form, i.e.,

$$\mathrm{F} = \mathrm{B} + \left(\mathrm{E} - \mathrm{D}\right)^{-1}. \tag{2.67}$$

Since a vector does not have a reciprocal, the reciprocal in equation (2.62) is replaced with a generalized inverse. The limit **F** of VEA, then, is calculated using

$$\mathbf{F} = \mathbf{B} + \left(\mathbf{E} - \mathbf{D}\right)^{+}, \tag{2.68}$$

where all variables are vectors. The generalized inverse chosen for this algorithm is written as

$$\left(\mathbf{E} - \mathbf{D}\right)^{+} = \frac{\left(\mathbf{E} - \mathbf{D}\right)}{\left\| \mathbf{E} - \mathbf{D} \right\|^{2}}, \tag{2.69}$$

where $\left\| \mathbf{E} - \mathbf{D} \right\|$ is the Euclidean norm. Note $\left(\mathbf{E} - \mathbf{D}\right)^{+}$ is a Moore-Penrose inverse, and its properties are presented in Appendix B. In Appendix B it is also proven that the generalized inverse given by equation (2.69) is indeed a Moore-Penrose inverse.

In summary, the operation of the first order VEA can be represented mathematically by Table 2.2 together with

$$\mathbf{D} = \mathbf{0} + \left(\mathbf{B} - \mathbf{A}\right)^{+}, \tag{2.70}$$

$$\mathbf{E} = \mathbf{0} + \left(\mathbf{C} - \mathbf{B}\right)^{+}, \tag{2.71}$$

and

$$\mathbf{F} = \mathbf{B} + \left(\mathbf{E} - \mathbf{D}\right)^{+}. \tag{2.72}$$

Graphically, the first order VEA fits a first order transient to three iterates, and then assumes the limit of the transient as an approximation to the limit of the original sequence. Note that for a single nonlinear equation, SEA and VEA are equivalent. Thus, for this thesis, the term "VEA" will be used when the algorithm is applied to either a single equation or a system of equations.

2.6.3 Hybrid Method

The Modified Newton's method (MNR) and the Vector Epsilon Algorithm (VEA) presented in the previous sections were combined to produce the hybrid method proposed in this thesis. The traditional Newton's method might be able to converge to a root quicker than other iterative methods, but with the possibility of overshooting the closest root relative to the initial guess. If a method cannot give the closest root, then there is a chance not all roots will be found. MNR in the hybrid method is able slow down the process such that it minimizes the chance of overshoot. Such a method is not effective in terms of rate of convergence, hence the VEA was needed to speed up the method.

In applying the first order VEA to MNR, the steps are as follows:

1. Pick an initial guess $\boldsymbol{x}_0$;
2. Produce 3 MNR iterates;
3. Use the 3 values obtained from MNR as the input to VEA;
4. Apply VEA as in Table 2.2 and equations (2.70) to (2.72);
5. Use the transform value **F** as the new initial guess for MNR;
6. Repeat steps 2 to 5 until consecutive MNR-VEA values are within a prescribed tolerance.

This thesis outlines numerical tests that were performed to gain a better understanding of the

method. In the literature, the behavior of VEA has not been entirely established, so more research is needed. Dolovich and Brodland [70] used VEA in accelerating iterative finite-element solution processes. Lowe [71] applied VEA in computerized tomography. Steele [9] found the set of iterative processes in one variable for which VEA extrapolates the iterates precisely to the limit (solution). This is called the kernel of first order VEA. However, the kernel of higher order VEA is still unknown. Also, the performance of VEA in obtaining approximations to a limit has not been established.

In the next chapter, the proposed hybrid MNR-VEA method was tested and compared to NR, Homotopy, and He's MNR (just described above) through a series of numerical experiments.

3 Numerical Experiments and Results

3.1 Introduction

The problems analyzed in this thesis are:

(i) One degree of freedom (DOF) problem (i.e., with one independent variable);

(ii) Two DOF problem (i.e., with two independent variables); and

(iii) Three DOF problem (i.e., with three independent variables).

The methods described in Chapter 2 were used to solve: Case (i), a single degree of freedom problem represented by a sinusoidal function which is commonly seen in engineering applications; Case (ii), a two degree of freedom problem involving two rigid links; and Case (iii), a three degree of freedom problem involving a robot arm with two rigid links and an extendable arm. A detailed description of each problem is given in sections to follow.

In this chapter, numerical experiments were performed on the nonlinear equations corresponding to Cases (i), (ii), and (iii). The results of the experiments are used as a guide to identify the performance of each iterative method (NR, Homotopy, HMNR, MNR-VEA). Customized codes written in the 2010 educational version of the numerical analysis software MATLAB® [12] were used to conduct the numerical simulations in MacBook Pro with 2.7 GHz Intel Core i5 processor and 8 GB memory. The problem details and results for Cases (i), (ii), and (iii) are discussed in Sections 3.2, 3.5, and 3.6, respectively.

3.2 One Degree of Freedom (1 DOF)

3.2.1 Detailed Problem Statement for 1 DOF

For one degree of freedom (DOF), the nonlinear equation $f(x) = 0$, where

$$f(x) = \sin x, \tag{3.1}$$

was used as a test case for this preliminary study. A graphical representation is given in Figure 3.1.

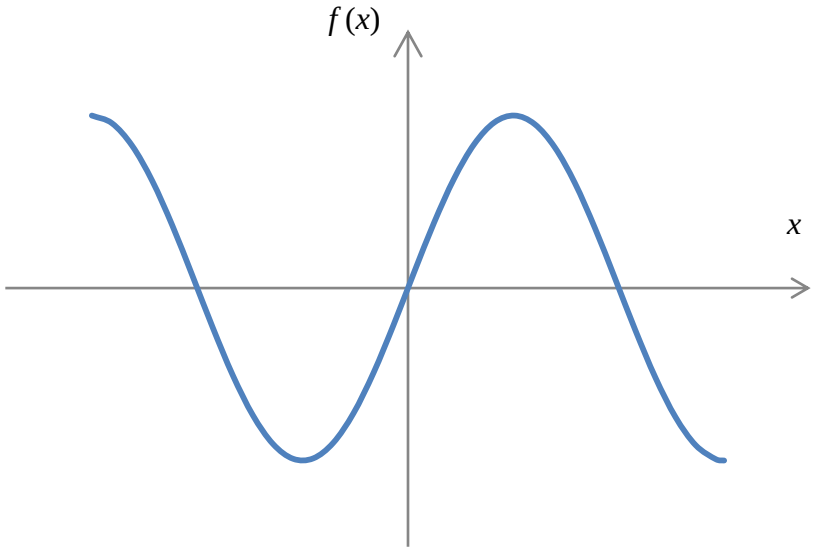

Figure 3.1: Graphical representation of nonlinear equation $f(x) = \sin x$

Equation (3.1) was chosen because it is fundamental to numerous engineering applications and, in particular, kinematic analyses where position vectors are defined by trigonometric functions. In addition, it is a simple equation with an analytical solution and therefore useful as the first test case. To test the performance of the iterative method, the initial guesses were selected specifically around certain points of interest, i.e.,

(a) near roots (where $f(x) = 0$);

(b) near critical points (where $f'(x)=0$); and

(c) in the region(s) between (a) and (b).

Regarding (a), the performance near roots is important since obtaining any particular solution to $f(x)=0$ is dependent on iterates converging to the root $x*$ once they are in its neighborhood (i.e., within $x*-\varepsilon<x<x*+\varepsilon$ for some ε). In this thesis, the neighborhood surrounding a root $x*$ is defined using the formulation

$$x^{(0)}-x*=-10^{-m}, \tag{3.2}$$

for *m*= 2, 3, and 4, to pick the initial guesses near the root $x*=\pi$ for the first test case. The initial guesses are shown in Table 3.1.

Table 3.1: Initial guesses near the root for first test case.

m	Initial Guess
2	3.131593
3	3.140593
4	3.141493

Regarding (b), the performance near critical points is important since the Newton's (NR) method tends to fail on providing the closest root precisely when an initial guess is near a critical point x^{C} of the function, where the slope is zero ($f'(x)=0$). In this thesis, the neighborhood surrounding a critical point x^{C} is defined using the formulation

$$x^{(0)}-x^{C}=10^{-m}, \tag{3.3}$$

for *m*= 2, 3, and 4, to pick the initial guesses near the critical point $x^{C}=\frac{\pi}{2}$. The initial guesses are shown in Table 3.2.

Regarding (c), the performance in the region between root $x*=\pi$ and critical point $x^{C}=\pi/2$ were also examined. The range of initial guesses in this region is $x^{C}+10^{-2}<x^{(0)}<x*-10^{-2}$. Both the

lower bound and the upper bound were rounded to 1 decimal place, and then initial guesses in this region were separated with equal increment of 0.1. Since all the initial guesses were between $\pi/2$ and π, the closest root for each initial guess was known to be π.

The total number of initial guesses for the first test case, equation (3.1), is 22. The performance of the four iterative methods (NR, Homotopy, HMNR, MNR-VEA), assessed via percentage differences from the solution, were evaluated using those initial guesses.

Table 3.2: Initial guesses near the critical point for first test case.

m	Initial Guess
2	1.580796
3	1.571796
4	1.570896

3.2.2 One DOF Results and Discussion of the Newton's Method (NR)

The results for NR are shown in Table 3.3. The first column gives initial guesses, the second column gives the solution. Referring to Table 3.3, the method failed to converge to the closest root when the initial guess was in the region $1.570896 \le x^{(0)} \le 1.9$ (indicated via shaded cells). For initial guesses close to the critical point $\pi/2$, the resulting solution overshot the closest root. For example, with an initial guess $x^{(0)} = 1.6$, the solution was 31.41593 as compared to the closest root of $\pi \approx 3.141593$. As the initial guesses approached the critical point, the overshooting increased, as seen in Table 3.3. As the initial guesses moved further from the critical point, the overshooting decreased until the initial guess achieved the closest root. The method converged to the closest root in the region $2 \le x^{(0)} \le 3.141493$, which can be attributed to the stopping criterions used in the calculations [11, 72]:

$$\| f(x_n) \| \leq \tau_r \| f(x_0) \| + \tau_a, \tag{3.4}$$

where the relative and absolute tolerance $\tau_r = \tau_a = 10^{-5}$; and

$$\left| x^{(k)} - x^{(k\text{-}1)} \right| < \delta, \tag{3.5}$$

where $\delta = 10^{-5}$.

Table 3.3: Newton's method results for $f(x) = \sin x$ (22 initial guesses).

Initial Guess	Solution
1.570896	9996.548
1.571796	1002.168
1.580796	100.531
1.6	31.41593
1.7	9.424778
1.8	6.283185
1.9	12.56637
2	3.141593
2.1	3.141593
2.2	3.141593
2.3	3.141593
2.4	3.141593
2.5	3.141593
2.6	3.141593
2.7	3.141593
2.8	3.141593
2.9	3.141593
3	3.141593
3.1	3.141593
3.131594	3.141593
3.140593	3.141593
3.141493	3.141593
Time (s)	0.008

A crucial factor to determine the performance of an iterative technique is the CPU time required for the iterative technique to converge to the closest root. This information provides insight into the performance of each method. The experiment ran 10 times and the minimum CPU time was 0.008s.

The overshooting phenomenon can be understood by considering Figure 3.3. If NR is used to solve the equation $f(x)=0$, where $f(x)=\sin x$, an initial guess $x^{(0)}$ close to the critical point (where $f'(x)\approx 0$) leads to a tangent line that has a small slope and intersects at a location $x^{(1)}$ far beyond the closest root. If an iterative method is to be considered stable, it should step towards the closest root without leaving the domain of interest.

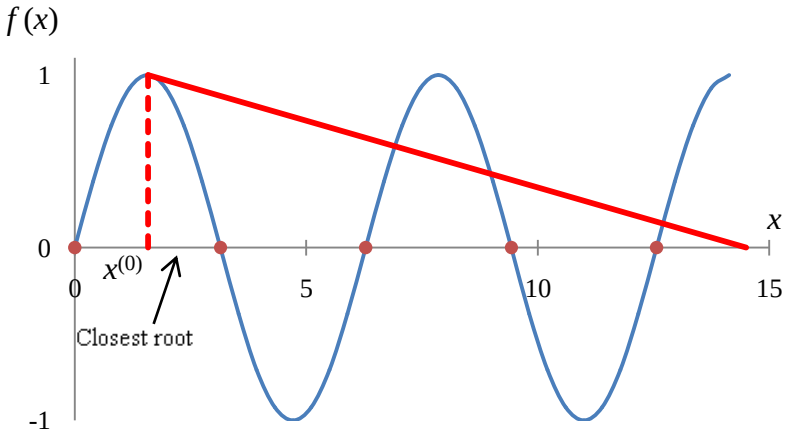

Figure 3.2: Graphical representation of overshooting for iterative method.

3.2.3 One DOF Results and Discussion of the Newton-Homotopy Method (Homotopy)

The results for the Homotopy method are shown in Table 3.4. The first column gives the initial guess, the rest have been divided into 4 sections which correspond to a different time increment, Δt, along with their solutions. For this test case, Δt values of 0.01, 0.02, 0.1, and 0.2 were selected. Results indicated that the Homotopy method was not affected by the initial guess (i.e., all initial guesses provided similar results for a chosen Δt value). The experiment ran 10 times and minimum CPU time for each Δt is listed in Table 3.4. The method required more computing

as time increments increased. Also, as expected, Homotopy provided more accurate answers with smaller Δt; though, Δt values of 0.02 and 0.01 provided similar accuracy while 0.02 were able to identify the solution in shorter time. According to the table, for all initial guesses within the domain $1.570896 \le x^{(0)} \le 3.141493$, the resulting solutions were able to converge to the closest root.

Table 3.4: Newton-Homotopy results for $f(x) = \sin x$(22 initial guesses).

	Solutions			
Initial Guesses	$\Delta t = 0.2$	$\Delta t = 0.1$	$\Delta t = 0.02$	$\Delta t = 0.01$
1.5708963	3.1561	3.144	3.1416	3.1416
1.5717963	3.1561	3.144	3.1416	3.1416
1.5807963	3.1561	3.144	3.1416	3.1416
1.6	3.1561	3.144	3.1416	3.1416
1.7	3.1561	3.144	3.1416	3.1416
1.8	3.1561	3.144	3.1416	3.1416
1.9	3.1561	3.144	3.1416	3.1416
2	3.1561	3.144	3.1416	3.1416
2.1	3.1561	3.144	3.1416	3.1416
2.2	3.1561	3.144	3.1416	3.1416
2.3	3.1561	3.144	3.1416	3.1416
2.4	3.1561	3.144	3.1416	3.1416
2.5	3.1561	3.144	3.1416	3.1416
2.6	3.1561	3.144	3.1416	3.1416
2.7	3.1561	3.144	3.1416	3.1416
2.8	3.1561	3.144	3.1416	3.1416
2.9	3.1561	3.144	3.1416	3.1416
3	3.1561	3.144	3.1416	3.1416
3.1	3.1561	3.144	3.1416	3.1416
3.1315926	3.1561	3.144	3.1416	3.1416
3.1405926	3.1561	3.144	3.1416	3.1416
3.1414926	3.1561	3.144	3.1416	3.1416
Time (s)	0.018	0.023	0.051	0.083

3.2.4 One DOF Results and Discussion of He's Modified Newton-Raphson Method (HMNR)

The results for HMNR are shown in Table 3.5. The first column gives the initial guess, the rest have been divided into 4 sections which correspond to a different *r*, along with their solutions.

For this test case, r values of 2, 3, 4, and 5 were selected. The shaded cells in Table 3.5 indicated the method did not converge to the true answer. The iterative technique ran 10 times and the minimum CPU time for each r is listed in Table 3.5. The computing time was 0.018s for r = 3, 4, and 5, and 0.199s for r = 2. Less computing time was required for larger r. The range of initial guesses for which the method failed to converge reduced as r increased. For example, the solutions for an initial guess $x^{(0)} = 1.6$ with r = 2, 3, 4 were unable to converge to the closest root; while, with r = 5, the solution was the true answer. The method became stable (and was able to achieve the closest root) with higher r and with an initial guess closer to the root.

Table 3.5: He's Modified Newton-Raphson results for $f(x) = \sin x$(22 initial guesses).

	Solutions			
Initial Guesses	$r = 2$	$r = 3$	$r = 4$	$r = 5$
1.5708963	172.7876	34.5575	18.8496	12.5664
1.5717963	53.4071	18.8496	12.5664	9.4248
1.5807963	18.8496	9.4248	6.2832	6.2832
1.6	15.708	6.2832	6.2832	3.1416
1.7	6.2832	3.1416	3.1416	3.1416
1.8	N/A	3.1416	3.1416	3.1416
1.9	3.1416	3.1416	3.1416	3.1416
2	3.1416	3.1416	3.1416	3.1416
2.1	3.1416	3.1416	3.1416	3.1416
2.2	3.1416	3.1416	3.1416	3.1416
2.3	3.1416	3.1416	3.1416	3.1416
2.4	3.1416	3.1416	3.1416	3.1416
2.5	3.1416	3.1416	3.1416	3.1416
2.6	3.1416	3.1416	3.1416	3.1416
2.7	3.1416	3.1416	3.1416	3.1416
2.8	3.1416	3.1416	3.1416	3.1416
2.9	3.1416	3.1416	3.1416	3.1416
3	3.1416	3.1416	3.1416	3.1416
3.1	3.1416	3.1416	3.1416	3.1416
3.1315926	3.1416	3.1416	3.1416	3.1416
3.1405926	3.1416	3.1416	3.1416	3.1416
3.1414926	3.1416	3.1416	3.1416	3.1416
Time (s)	0.199	0.018	0.018	0.018

3.2.5 One DOF Results and Discussion of Modified Newton's Method with the Vector Epsilon Algorithm (MNR-VEA)

The results for MNR-VEA are shown in Table 3.6.The first column gives the initial guess, the rest have been divided into 4 sections which correspond to a different α, along with their solutions. Table 3.6 also illustrates the time required for the iterative technique as α increased. For this test case, α values of 0.1, 0.01, 0.001, and 0.0001 were selected. The shaded cells in Table 3.6 indicate the method did not converge to the true answer with α values of 0.1, 0.01 and 0.001. The range of initial guesses for which the method failed to converge was reduced as α decreased. With $\alpha = 0.0001$, all initial guesses converged. This was expected as MNR-VEA required longer computation time since it is a modified Newton's methods with a relaxation parameter.

Table 3.6: MNR-VEA results for $f(x) = \sin x$(22 initial guesses).

	Solutions			
Initial Guesses	$\alpha = 0.1$	$\alpha = 0.01$	$\alpha = 0.001$	$\alpha = 0.0001$
1.5708963	1002.168	100.531	12.5664	3.1416
1.5717963	100.531	12.5664	3.1416	3.1416
1.5807963	12.5664	3.1416	3.1416	3.1416
1.6	6.2832	3.1416	3.1416	3.1416
1.7	3.1416	3.1416	3.1416	3.1416
1.8	3.1416	3.1416	3.1416	3.1416
1.9	3.1416	3.1416	3.1416	3.1416
2	3.1416	3.1416	3.1416	3.1416
2.1	3.1416	3.1416	3.1416	3.1416
2.2	3.1416	3.1416	3.1416	3.1416
2.3	3.1416	3.1416	3.1416	3.1416
2.4	3.1416	3.1416	3.1416	3.1416
2.5	3.1416	3.1416	3.1416	3.1416
2.6	3.1416	3.1416	3.1416	3.1416
2.7	3.1416	3.1416	3.1416	3.1416
2.8	3.1416	3.1416	3.1416	3.1416
2.9	3.1416	3.1416	3.1416	3.1416
3	3.1416	3.1416	3.1416	3.1416
3.1	3.1416	3.1416	3.1416	3.1416
3.1315926	3.1416	3.1416	3.1416	3.1416
3.1405926	3.1416	3.1416	3.1416	3.1416
3.1414926	3.1416	3.1416	3.1416	3.1416
Time (s)	0.024	0.024	0.024	0.024

3.3 Summary of 1 DOF

NR, HMNR and some cases of MNR-VEA (with α values of 0.1, 0.01 and 0.001) failed to converge on closest root specifically when the initial guesses were in the region close to the critical point of 1.5708. For all values of Δt, Homotopy were able to converge to the desired root; MNR-VEA, with an α value of 0.0001, did not fail to converge. HMNR with r = 2 took the longest time when compared to the other methods. As expected, NR was the fastest out of the four methods.

3.4 Degree of Freedom (2 DOF)

3.4.1 Detailed Problem Statement for 2 DOF

For the two degree of freedom test case, the system of nonlinear equations is $\boldsymbol{f}(\boldsymbol{x}) = \mathbf{0}$, where

$$f_1(\beta,\gamma) = \mathrm{L}_1 \cos\beta + \mathrm{L}_2 \cos\gamma - x_\mathrm{P}, \tag{3.6}$$

$$f_2(\beta,\gamma) = \mathrm{L}_1 \sin\beta + \mathrm{L}_2 \sin\gamma - y_\mathrm{P}, \tag{3.7}$$

and these equations represent a robot arm with two links as given in Figure 3.7. The base is fixed, and the links have lengths L_1 = 2 and L_2 = 1 where, similarly to Wu [73], units are not specified since they would not affect the simulations and results to be considered. One end of link L_1 is connected to pin O with the other end pin-connected to link L_2 at B. Gripper P is at the end of the robot arm, and it needs to pass through coordinates (x_P, y_P) = (-1, 1.5). The unknowns are β and γ, and they are measured from the positive x-axis to the robot links.

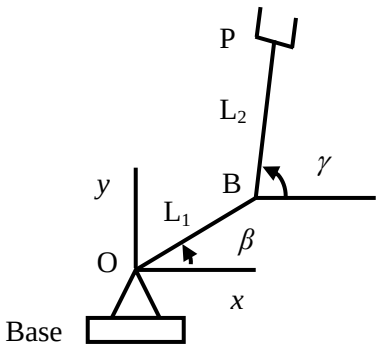

Figure 3.3: Robot linkage, Two Degree of Freedom (After [73]).

In this research, angles β and γ were varied from 0° to 360°, and with an increment of 10° for each angle (e.g., 0°, 10°, 20° …). In total, there were 1369 guesses (e.g., 37 β angles × 37 γ angles = 1369 combinations).

For the one DOF case, the single function $f(x)$ can be represented by a curve in a 2-D plot (i.e., $f(x)$ vs x). However, for the two DOF case, each of the functions $\boldsymbol{f}_i(\boldsymbol{x}_j)$ must be represented by a surface. An alternate approach to sketching these surfaces is to first write $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ as

$$\mathrm{L}_1 \cos\beta + \mathrm{L}_2 \cos\gamma - x_\mathrm{P} = 0, \tag{3.8}$$

$$\mathrm{L}_1 \sin\beta + \mathrm{L}_2 \sin\gamma - y_\mathrm{P} = 0. \tag{3.9}$$

Although (3.8) and (3.9) are both equations, each may alternatively be regarded as giving γ as a function of β implicitly. Using MATLAB® [12], each of these implicit functions (i.e., one function for (3.8) and one function for (3.9)) have been plotted in Figure 3.8 for $0^\mathrm{o} \le \beta \le 360^\mathrm{o}$ and $0^\mathrm{o} \le \gamma \le 360^\mathrm{o}$. The roots to $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$ are located at points where the two functions intersect (i.e., where (3.8) and (3.9) are both satisfied).

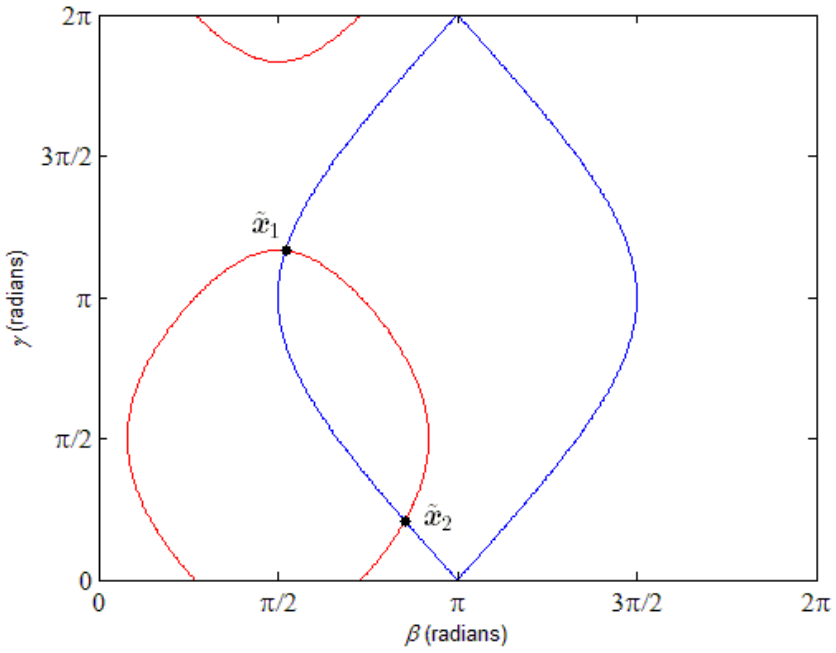

Figure 3.4: Graphical representation of roots of $f(x) = 0$ with $f_i(x_j)$ given by equations (3.6) and (3.7), for β from 0 to 2π and γ from 0 to 2π. Blue corresponds to equation (3.6); red corresponds to (3.7).

Solutions within the domain ($0^o \le \beta \le 360^o$ and $0^o \le \gamma \le 360^o$) are denoted as $\tilde{\boldsymbol{x}}_i$, and there are two solutions for this two DOF test case:

$$\tilde{\boldsymbol{x}}_1 = \left(\tilde{\beta}_1, \tilde{\gamma}_1\right) \approx \left(1.6366\,\text{rad}, 3.6601\,\text{rad}\right), \tag{3.10}$$

$$\tilde{\boldsymbol{x}}_2 = \left(\tilde{\beta}_2, \tilde{\gamma}_2\right) \approx \left(2.6810\,\text{rad}, 0.6564\,\text{rad}\right). \tag{3.11}$$

They can be expressed in terms of degrees:

$$\tilde{\boldsymbol{x}}_1 = \left(\tilde{\beta}_1, \tilde{\gamma}_1\right) \approx \left(93.77^o, 209.71^o\right), \tag{3.12}$$

$$\tilde{\boldsymbol{x}}_2 = \left(\tilde{\beta}_2, \tilde{\gamma}_2\right) \approx \left(153.61^o, 37.67^o\right). \tag{3.13}$$

A given numerical method (i.e., NR, Homotopy, HMNR or MNR-VEA) can obtain a solution $\hat{\boldsymbol{x}}$ which is not necessarily within the domain. That solution is checked by placing it back into the

equations to ensure it is a root, i.e., equations (3.6) and (3.7). However, it does not guarantee it is the closest root.

For a numerical method in the two DOF test case, the closest root $\boldsymbol{x}^*$ to an initial guess $\boldsymbol{x}^{(0)} = \left(\beta^{(0)}, \gamma^{(0)}\right)$ is the $\tilde{\boldsymbol{x}}_i$ with the smallest distance to $\boldsymbol{x}^{(0)}$, where the distance $\tilde{\rho}_i$ is defined as

$$\tilde{\rho}_i = \left\| \tilde{\boldsymbol{x}}_i - \boldsymbol{x}^{(0)} \right\|, \tag{3.14}$$

$$\tilde{\rho}_i = \sqrt{\left(\tilde{\beta}_i - \beta^{(0)}\right)^2 + \left(\tilde{\gamma}_i - \gamma^{(0)}\right)^2}\,. \tag{3.15}$$

For a given $\boldsymbol{x}^{(0)}$, the closest root $\boldsymbol{x}^*$ is that particular root $\tilde{\boldsymbol{x}}_i$ where $\tilde{\rho}_i$ is minimum. That is, for a given initial guess $\boldsymbol{x}^{(0)}$, the closest root is identified by calculating $\tilde{\rho}_1$ for $\tilde{\boldsymbol{x}}_1$, and $\tilde{\rho}_2$ for $\tilde{\boldsymbol{x}}_2$. If $\tilde{\rho}_1 < \tilde{\rho}_2$, then $\tilde{\boldsymbol{x}}_1 = \boldsymbol{x}^*$, and vice versa. For any calculated solution $\hat{\boldsymbol{x}}$ from a given method using an initial guess $\boldsymbol{x}^{(0)}$, the distance between that $\hat{\boldsymbol{x}}$ and $\boldsymbol{x}^*$ is defined as

$$\rho = \left\| \hat{\boldsymbol{x}} - \boldsymbol{x}^* \right\|, \tag{3.16}$$

$$\rho = \sqrt{\left(\hat{\beta} - \beta^*\right)^2 + \left(\hat{\gamma} - \gamma^*\right)^2}\,. \tag{3.17}$$

This is illustrated in Figure 3.9. In this figure, the closest root $\boldsymbol{x}^*$ for the given initial guess $\boldsymbol{x}^{(0)}$ is $\tilde{\boldsymbol{x}}_1$ since $\tilde{\rho}_1 < \tilde{\rho}_2$. However, the numerical method gives $\hat{\boldsymbol{x}}$ (in this case, a solution which is not within the domain) as the solution which is neither $\tilde{\boldsymbol{x}}_1$ nor $\tilde{\boldsymbol{x}}_2$. The performance of the numerical method is measured by value ρ (the smaller ρ, the better the performance because it is closer to the desired root). A small ρ indicates the calculated solution is close to the closest root, i.e., when $\rho = 0$, the method yielded the closest root.

Ideally, for every initial guess, a numerical method would give $\hat{\boldsymbol{x}}$ such that $\rho = 0$. In practice, methods are compared by their ability to consistently achieve small values of ρ.

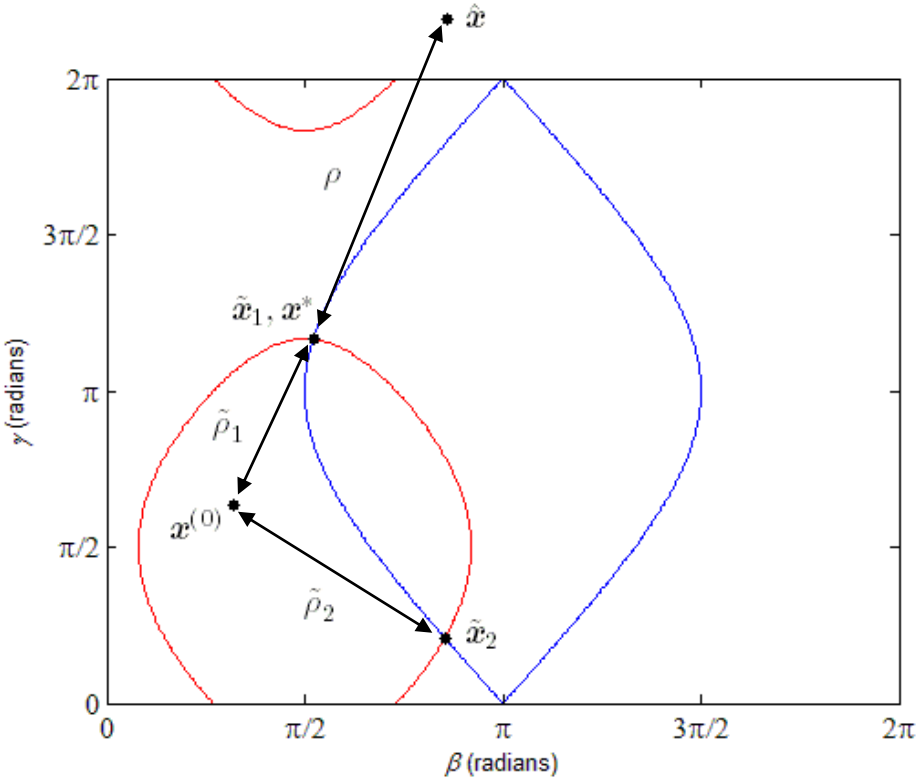

Figure 3.5: Graphical representation of roots of $f(x) = 0$ with $f_i(x_j)$ given by equations (3.6) and (3.7), for β from 0 to 2π and γ from 0 to 2π. Blue corresponds to equation (3.6) and red corresponds to equation (3.7). This demonstrates different parameters using the 2 DOF case.

3.4.2 Two DOF Results and Discussion of the Newton's method (NR)

A summary of the results for the NR method is shown in Table 3.7. The termination criterion for 2 DOF problem is equation (**2.2**) with $\tau_r = \tau_a = 10^{-5}$.

Table 3.7: Summary of NR results for the 2 DOF test case (1369 initial guesses). In this table, the first column gives the CPU time; second gives the percentage of initial guesses giving valid solutions (defined as a residual less than 0.03394); third column gives the percentage of initial guesses yielding $\rho = 0$; fourth column gives the average value of ρ (for the valid solutions); and the last column gives the maximum value of ρ (for the valid solutions).

CPU Time (s)	Percentage of initial guesses giving valid solutions	Percentage of initial guesses yielding $\rho = 0$	ρ_{avg}(radians)	ρ_{max} (radians)
1.547	94%	30%	1.1021×10^{15}	1.7822×10^{12}

For this test case, the CPU time was 1.547s, and out of the 1369 initial guesses, 1293 initial guesses gave valid solutions. For this analysis, a "valid solution" was defined as a calculated result which, when substituted into the equations, gave a residual less than 0.03394; i.e., $\|\boldsymbol{f}(\boldsymbol{x}_i)\| < 0.03394$. The threshold value of "0.03394" was obtained by calculating $\|\boldsymbol{f}(\boldsymbol{x}_i)\|$ for different combinations of $\tilde{\boldsymbol{x}}_1$ and $\tilde{\boldsymbol{x}}_2$ where β values were within $\tilde{\beta} \pm 0.1^\circ$ and γ values were within $\tilde{\gamma} \pm 0.1^\circ$; for this range, the maximum value of $\|\boldsymbol{f}_i(\tilde{\boldsymbol{x}}_j)\|$ was 0.03394. (In this exercise of finding a reasonable threshold value for any root, $\tilde{\beta}$ and $\tilde{\gamma}$ were just used as two known solutions to compare variations from a root to variations in the residual.) Of the 1293 valid solutions, 406 were the closest root. However, the average ρ and max ρ showed that, in many cases, the method gave solutions far outside the domain.

3.4.3 Two DOF Results and Discussion of Newton-Homotopy method (Homotopy)

A summary of the results for the Homotopy method are shown in Table 3.8.

Table 3.8: Summary of Homotopy results for the 2 DOF test case (1369 initial guesses). In this table, the first column gives different values of Δt, the second column gives the CPU time, the third column gives the percentage of of initial guesses giving valid solutions, the fourth column gives the percentage of initial guesses yielding $\rho = 0$, the fifth column gives the average value of ρ (for the valid solutions), and the last column gives the maximum value of ρ (for the valid solutions).

Δt	CPU Time (s)	Percentage of initial guesses giving valid solutions	Percentage of initial guesses yielding $\rho = 0$	ρ_{avg}(radians)	ρ_{max} (radians)
0.2	13.637	75%	40%	5.1261×10^{11}	3.2345×10^{14}
0.1	26.554	74%	40%	1.7947×10^{12}	5.2048×10^{14}
0.02	131.280	59%	40%	3.3753×10^{12}	1.4494×10^{15}
0.01	267.880	59%	40%	1.5114×10^{12}	3.2186×10^{14}

For this test case, Δt values of 0.2, 0.1, 0.02, and 0.01 were selected. Less than half of the initial guesses were able to provide the closest root. Interestingly, the number of initial guesses yielding valid solutions decreased with smaller Δ*t* and the CPU time increased. Though, all Δ*t* gave the exact same percentage of initial guesses yielding $\rho = 0$. Again, like the NR method, the average ρ and the maximum ρ values indicated the existence of solutions which were far outside the domain.

3.4.4 Two DOF Results and Discussion of He's Modified Newton-Raphson method (HMNR)

A summary of results for the HMNR method are shown in Table 3.9. This method gives complex numbers because of the nature of the equations where powers are involved, which results in square roots and cubic roots, etc.

Table 3.9: Summary of HMNR results for the 2 DOF test case (1369 initial guesses). In this table, the first column gives different values of *r*; second column gives the CPU time; third column gives the percentage of initial guesses giving valid solutions; fourth column gives the percentage of initial guesses yielding $\rho = 0$; fifth column gives the number of complex number; sixth column gives the average value of ρ (for the valid solutions); the last column gives the maximum value of ρ (for the valid solutions).

r	CPU Time (s)	Percentage of initial guesses giving valid solutions	Percentage of initial guesses yielding $\rho = 0$	Number of Complex Number	ρ_{avg}(radians)	ρ_{max} (radians)
2	2.714	24%	15%	977	9.4536×10^{6}	3.5814×10^{8}
3	5.852	15%	11%	1106	1.0060×10^{5}	6.6836×10^{5}
4	6.284	12%	9%	1152	153.498	2.4826×10^{4}
5	10.039	10%	9%	1184	18.2628	2.3750×10^{3}

For this test case, *r* values of 2, 3, 4, and 5 were selected. With HMNR, less time was required for convergence and ρ was low for valid solutions (Table 3.9). However, the number of initial

guesses yielding $\rho = 0$ and the number of valid solutions decreased with higher r. Accordingly, increasing r did not improve performance, at least for this test case.

3.4.5 Two DOF Results and Discussion of Modified Newton's Method with the Vector Epsilon Algorithm (MNR-VEA)

A summary of the results for the MNR-VEA method are shown in Table 3.10.

Table 3.10: Summary of MNR-VEA results for the 2 DOF test case (1369 initial guesses). In this table, the first column gives different values of α, the second column gives the CPU time, the third column gives the percentage of initial guesses giving valid solutions, the fourth column gives the percentage of initial guesses yielding $\rho = 0$, the fifth column gives the average value of ρ (for the valid solutions), and the last column gives the maximum value of ρ (for the valid solutions).

α	CPU Time (s)	Percentage of initial guesses giving valid solutions	Percentage of initial guesses yielding $\rho = 0$	ρ_{avg} (radians)	ρ_{max} (radians)
0.1	2.066	94%	46%	5.7763	1492.413
0.01	2.663	94%	44%	2.7072	23.3116
0.001	2.696	94%	44%	2.6843	22.1546
0.0001	2.763	94%	45%	2.6287	12.0481

For this test case, α values of 0.1, 0.01, 0.001, and 0.0001 were selected. The longest CPU time corresponded with $\alpha = 0.0001$. For all α, 1288 of 1369 initial guesses give valid solutions. The values for the average ρ and the maximum ρ decreased with smaller α. Thus, the accuracy of achieving the closest root increased as α decreased. The number of valid solutions remained the same as α decreased. Although there is not a clear pattern between the number of initial guesses yielding $\rho = 0$ and α, in general, approximately 45% of the initial guesses provided the closest root, which was slightly better than the Homotopy results of 40%.

3.4.6 Summary of 2 DOF

For the 2 DOF test case, the success rate of all numerical methods in finding the closest root was under 50%. HMNR did poorly in arriving at the closest root because of the nature of the method (square roots and cubic roots etc., which resulted in complex numbers). Changing the value of r did not yield better results. Homotopy was consistent in arriving at the closest root; however, the average ρ and maximum ρ were large, indicating solutions outside the domain. The frequency of obtaining the closest root increased as Δt decreased. MNR-VEA has a better performance in converging to the closest root with the shortest computing time, and was more consistent in obtaining valid solutions, as summarized in Table 3.11.

Table 3.11: Summary of the 2 DOF test case (1369 initial guesses).

Methods		CPU Time (s)	Percentage of initial guesses yielding $\rho = 0$	Percentage of valid solutions	ρ_{avg} (radians)	ρ_{max} (radians)
NR		1.547	94%	30%	1.1021×10^{15}	1.7822×10^{12}
Homotopy (Δt)	0.2	13.637	75%	40%	5.1261×10^{11}	3.2345×10^{14}
	0.1	26.554	74%	40%	1.7947×10^{12}	5.2048×10^{14}
	0.02	131.280	59%	40%	3.3753×10^{12}	1.4494×10^{15}
	0.01	267.880	59%	40%	1.5114×10^{12}	3.2186×10^{14}
HMNR (r)	2	2.714	24%	15%	9.4536×10^{6}	3.5814×10^{8}
	3	5.852	15%	11%	1.0060×10^{5}	6.6836×10^{5}
	4	6.284	12%	9%	153.498	2.4826×10^{4}
	5	10.039	10%	9%	18.2628	2.3750×10^{3}
MNR-VEA (α)	0.1	2.066	94%	46%	5.7763	1492.413
	0.01	2.663	94%	44%	2.7072	23.3116
	0.001	2.696	94%	44%	2.6843	22.1546
	0.0001	2.763	94%	45%	2.6287	12.0481

3.4.7 Converting a 2 DOF to a 1 DOF Problem

Performances of numerical methods for the 2 DOF test case did not necessarily meet expectations. As an alternative, the 2 DOF problem can be converted into a 1 DOF test case, which might result in improved performance. Squaring and adding equations (3.8) and (3.9) gives

$$4\cos\beta - 6\sin\beta + 6.25 = 0. \tag{3.18}$$

(Appendix C gives the derivation of Equation (3.18), as well as the analytical solution for the roots in the domain). Equation (3.18) is now a 1 DOF problem with β the only unknown. This test case, corresponding to equation (3.18), will be denoted the "2-to-1 DOF test case" to distinguish it from the 1 DOF test case presented in Section 3.2. A graphical representation of the corresponding function $f(x) = 4\cos x - 6\sin x + 6.25$ is given in Figure 3.10. Here β was varied from -60° to 400°, in increment of 10 degrees, resulting in 47 initial guesses.

For this case, the closest root, $x^* = \beta^*$, to a particular initial guess, $x^{(0)} = \beta^{(0)}$, can be found by calculating the distance $\tilde{\rho}_i$ between $\beta^{(0)}$ and each root $\tilde{\beta}_i$ in the domain, and finding the minimum value of $\tilde{\rho}_i$. That is, x^* is determined by discrete minimization of $\tilde{\rho}_i$, where

$$\tilde{\rho}_i = \left|\tilde{\beta}_i - \beta^{(0)}\right|. \tag{3.19}$$

For any calculated solution $\hat{x} = \hat{\beta}$ from a given method using an initial guess $x^{(0)}$, the distance between that $\hat{x}$ and x^* is defined as

$$\rho = \left|\hat{\beta} - \beta^*\right|. \tag{3.20}$$

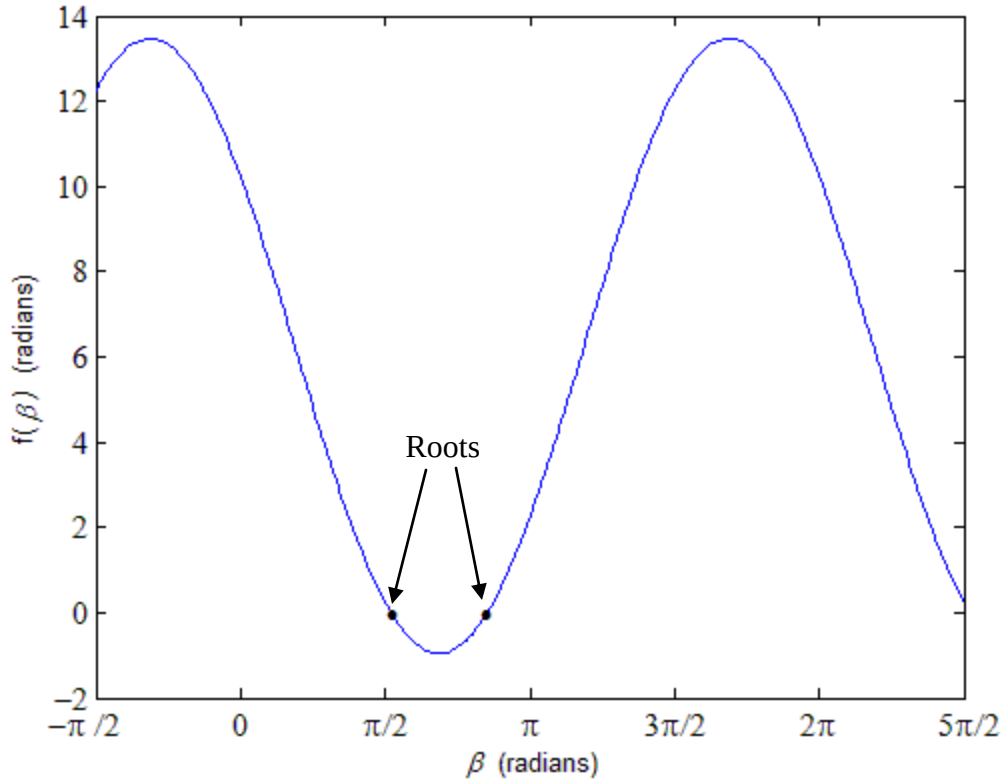

Figure 3.6: Graphical representation $f(\beta) = 0$ given by equations (3.18).

Since a part of the MNR-VEA is a modified version of Newton's method, if Newton's method does not go in the direction of the closest root in the domain, $x*$, it is expected that MNR-VEA will not change the direction of the process to obtain $x*$. However, it might be that MNR-VEA can assist in finding $x**$, which is defined here as the closest root (to the initial guess) in the direction defined by the tangent used in NR, regardless of whether that root is within the domain. This possibility is evaluated by the values in Table 3.12. Results indicated that the frequency of achieving *x*** increased as *r* increased. MNR-VEA had the best performance in achieving *x** and *x***.

Table 3.12: Summary of results for the 2/1 DOF test case (47 initial guesses).

Methods		Percentage of initial guesses yielding x^*	Percentage of initial guesses yielding x^{**}
NR		98%	87%
Homotopy (Δt)	0.2	94%	94%
	0.1	94%	94%
	0.02	94%	94%
	0.01	94%	94%
HMNR (r)	2	98%	85%
	3	98%	94%
	4	98%	98%
	5	98%	98%
MNR-VEA (α)	0.1	100%	100%
	0.01	100%	100%
	0.001	100%	100%
	0.0001	100%	100%

3.5 Three Degree of Freedom (3 DOF)

3.5.1 Detailed Problem Statement for 3 DOF

For the three degree of freedom test case, the system of nonlinear equations is $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$, where

$$f_1(\beta,\gamma,c) = -\mathrm{L}_2 \cos\beta \sin\gamma + a - x_P, \tag{3.21}$$

$$f_2(\beta,\gamma,c) = \mathrm{L}_2 \cos\gamma + c + \mathrm{L}_1 - y_P, \tag{3.22}$$

$$f_3(\beta,\gamma,c) = \mathrm{L}_2 \sin\beta \sin\gamma - b - z_P, \tag{3.23}$$

which represent a robot arm with a gripper on a fixed pole as given in Figure 3.11. The base of the pole is located such that a = 104 and b = 7, and the length of pole is L_1 = 60. The robot arm has a length of L_2 = 25 and it is free to rotate about the y'-axis. Units are not specified since they would not affect the simulations and results to be considered. Gripper P is at the end of the robot

arm, and it needs to pass through coordinates $(x_P, y_P, z_P) = (100, 95, -30)$. The unknowns are β, γ and c, where β and γ are the angles associated with the robot arm and c is the changeable pole height.

In this research, angles β and γ were varied from 0° to 360° with an increment of 10° for each angle; the changeable height of the pole, c, varied from 0 to 60 with an increment of 5. There were 37 × 37 × 13 = 17797 guesses in total.

When the program arrived at a result, it checked this result by placing it back into the equations, i.e., equations (3.21), (3.22) and (3.23).

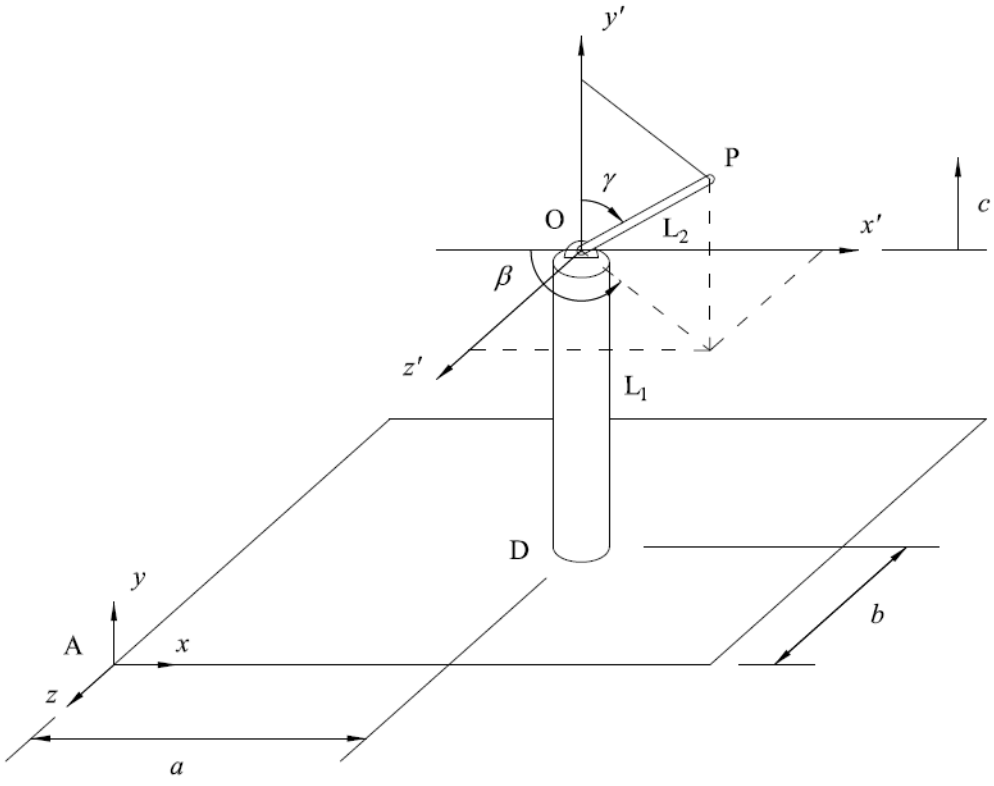

Figure 3.7: Robot linkage, Three Degree of Freedom (After [54]).

Solutions within the domain ($0^o \le \beta \le 360^o, 0^o \le \gamma \le 360^o$, and $0 \le c \le 60$) are denoted as $\tilde{\boldsymbol{x}}_i$, and there are four solutions [54] for this three DOF test case:

$$\tilde{\boldsymbol{x}}_1 = \left(\tilde{\beta}_1, \tilde{\gamma}_1, c\right) \approx (1.7431 \text{ rad}, 4.3466 \text{ rad}, 43.94), \tag{3.22}$$

$$\tilde{\boldsymbol{x}}_2 = \left(\tilde{\beta}_2, \tilde{\gamma}_2, c\right) \approx (1.7431 \text{ rad}, 3.6645 \text{ rad}, 26.06), \tag{3.26}$$

$$\tilde{\boldsymbol{x}}_3 = \left(\tilde{\beta}_2, \tilde{\gamma}_2, c\right) \approx (4.8847 \text{ rad}, 1.2050 \text{ rad}, 26.06), \tag{3.27}$$

$$\tilde{\boldsymbol{x}}_4 = \left(\tilde{\beta}_2, \tilde{\gamma}_2, c\right) \approx (4.8847 \text{ rad}, 1.9366 \text{ rad}, 43.94). \tag{3.28}$$

They can be expressed in terms of degrees:

$$\tilde{\boldsymbol{x}}_1 = \left(\tilde{\beta}_1, \tilde{\gamma}_1, \tilde{c}\right) \approx (99.87^o, 249.04^o, 43.94), \tag{3.29}$$

$$\tilde{\boldsymbol{x}}_2 = \left(\tilde{\beta}_2, \tilde{\gamma}_2, \tilde{c}\right) \approx (99.87^o, 209.96^o, 26.06), \tag{3.30}$$

$$\tilde{\boldsymbol{x}}_3 = \left(\tilde{\beta}_2, \tilde{\gamma}_2, \tilde{c}\right) \approx (279.87^o, 69.04^o, 26.06), \tag{3.31}$$

$$\tilde{\boldsymbol{x}}_4 = \left(\tilde{\beta}_2, \tilde{\gamma}_2, \tilde{c}\right) \approx (279.87^o, 110.96^o, 43.94). \tag{3.32}$$

For a numerical method in the 3 DOF test case, the closest root $\boldsymbol{x}^*$ to an initial guess $\boldsymbol{x}^{(0)} = \left(\beta^{(0)}, \gamma^{(0)}, c^{(0)}\right)$ is the $\tilde{\boldsymbol{x}}_{\boldsymbol{i}}$ with the smallest distance to $\boldsymbol{x}^{(0)}$, where the distance $\tilde{\rho}_i$ is defined as

$$\tilde{\rho}_i = \sqrt{\left(\frac{\tilde{\beta}_i - \beta^{(0)}}{\beta_{max} - \beta_{min}}\right)^2 + \left(\frac{\tilde{\gamma}_i - \gamma^{(0)}}{\gamma_{\max} - \gamma_{\min}}\right)^2 + \left(\frac{\tilde{c} - c^{(0)}}{c_{\max} - c_{\min}}\right)^2}, \tag{3.33}$$

where $\beta_{min} \le \beta \le \beta_{max}$, $\gamma_{\min} \le \gamma \le \gamma_{\max}$, $c_{\min} \le c \le c_{\max}$ and $\beta_{max} = \gamma_{max} = 2\pi$, $\beta_{min} = \gamma_{min} = 0$, $c_{\max} = 60, c_{\min} = 0$. For any calculated solution $\hat{\boldsymbol{x}}$ from a given method using an initial guess $\boldsymbol{x}^{(0)}$, the distance between that $\hat{\boldsymbol{x}}$ and $\boldsymbol{x}^*$ is defined as

$$\rho = \sqrt{\left(\frac{\hat{\beta} - \beta^*}{\beta_{max} - \beta_{min}}\right)^2 + \left(\frac{\hat{\gamma} - \gamma^*}{\gamma_{\max} - \gamma_{\min}}\right)^2 + \left(\frac{\hat{c} - c^*}{c_{\max} - c_{\min}}\right)^2} \quad . \tag{3.34}$$

3.5.2 Three DOF Results and Discussion of the Newton's method (NR)

A summary of the results for the NR method are shown in Table 3.13. The termination criterion for 3 DOF problem is equation(2.2) with $\tau_r = \tau_a = 10^{-5}$. For the purpose of this analysis, a "valid solution" is defined as a calculated result which, when substituted into the equations, gives a residual less than 0.04385, which correspond with a maximum value of $\boldsymbol{f}_i(\boldsymbol{x}) = 0.1436$. The threshold value of "0.1436" was obtained by calculating $\max_i |\boldsymbol{f}_i(\boldsymbol{x})|$ for different combinations of $\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \tilde{\boldsymbol{x}}_3$ and $\tilde{\boldsymbol{x}}_4$ where β values were within $\tilde{\beta} \pm 0.1^\circ$, γ values were within $\tilde{\gamma} \pm 0.1^\circ$ and c values were within $\tilde{c} \pm 0.1$.

Table 3.13: Summary of NR results for the 3 DOF test case (17797 initial guesses). In this table, the first column gives the CPU time; second column gives the percentage of initial guesses giving valid solutions; third column gives the percentage of initial guesses yielding $\rho = 0$; fourth column gives the average value of ρ (for valid solutions); the last column gives the maximum value of ρ (for valid solutions).

CPU Time (s)	Percentage of initial guessing yielding valid solutions	Percentage of initial guesses yielding $\rho = 0$	ρ_{avg} (radians)	$\rho_{\max}$ (radians)
35	87%	54%	2.0949×10^{11}	1.7205×10^{14}

For this test case, out of the 17797 initial guesses, 15466 initial guesses gave valid solutions while 9660 initial guesses gave the closest root with $\rho = 0$. The average ρ and max ρ values were large, indicating that in many cases the method gave solutions far outside the domain.

3.5.3 Three DOF Results and Discussion of Newton-Homotopy method (Homotopy)

A summary of the results for the Homotopy method are shown in Table 3.14. For this test case, Δt values of 0.2, 0.1, 0.02, and 0.01 were selected. The computing time increased as more time increments were used. The majority of the initial guesses were able to provide the closest root (~83% of cases). The number of initial guesses yielding the closest root appeared to be independent of the choice of Δt. The average ρ and the maximum ρ values indicate the existence of solutions far outside the domain.

Table 3.14: Summary of Homotopy results for the 3 DOF test case (17797 initial guesses). In this table, the first column gives different values of Δt, the second column gives the CPU time, the third column gives the percentage of initial guesses giving valid solutions, the fourth column gives the percentage of initial guesses yielding $\rho = 0$, the fifth column gives the average value of ρ (for the valid solutions), and the last column gives the maximum value of ρ (for the valid solutions).

Δt	CPU Time (s)	Percentage of initial guesses yielding valid solutions	Percentage of initial guesses yielding $\rho = 0$	ρ_{avg} (radians)	ρ_{max} (radians)
0.2	32	87%	81%	3.5995×10^{11}	1.4555×10^{14}
0.1	42	87%	83%	4.5332×10^{11}	1.6260×10^{14}
0.02	127	93%	84%	8.6112×10^{11}	7.6354×10^{13}
0.01	231	95%	84%	6.7130×10^{11}	4.8780×10^{13}

3.5.4 Three DOF Results and Discussion of He's Modified Newton-Raphson method (HMNR)

Due to the nature of how HMNR iterates (via equations (3.35) to (3.37) located below), the method was unable to solve the problem. Specifically, equation (3.37) involved division by zero. As such, there was no solution for the 3 DOF problem using HMNR.

$$\beta_{n+1}^{r} = \beta_{n}^{r} - \frac{r\,\beta_{n}^{r-1} f_1(\beta_n)}{f_1'(\beta_n)}\,, \tag{3.35}$$

$$\gamma_{n+1}^{r} = \gamma_{n}^{r} - \frac{r\,\gamma_{n}^{r-1} f_2(\gamma_n)}{f_2'(\gamma_n)}\ , \tag{3.36}$$

$$c_{n+1}^{r} = c_{n}^{r} - \frac{r\,c_{n}^{r-1} f_3(c_n)}{f_3'(c_n)}\ , \tag{3.37}$$

3.5.5 Three DOF Results and Discussion of Modified Newton's Method with the Vector Epsilon Algorithm (MNR-VEA)

A summary of the results for the MNR-VEA method are shown in Table 3.15.

Table 3.15: Summary of MNR-VEA results for the 3 DOF test case (17797 initial guesses). In this table, the first column gives different values of the relaxation parameter, α, the second column gives the CPU time, the third column gives the percentage of initial guesses giving valid solutions, the fourth column gives the percentage of initial guesses yielding $\rho = 0$, the fifth column gives the average value of ρ (for the valid solutions), and the last column gives the maximum value of ρ (for the valid solutions).

α	CPU Time (s)	Percentage of initial guesses yielding valid solutions	Percentage of initial guesses yielding $\rho = 0$	ρ_{avg} (radians)	ρ_{max} (radians)
0.1	1165	85%	3%	29.2098	14647.5
0.01	3608	48%	11%	1.97745	567.754
0.001	3360	51%	18%	0.52514	23.6008
0.0001	3156	50%	17%	0.55197	86.0349

For this test case, α values of 0.1, 0.01, 0.001, and 0.0001 were selected. The CPU time, number of initial guesses yielding $\rho = 0$ and the number of valid solutions did not have a clear pattern with variations of α. The required time for $\alpha < 0.1$ tripled when compared to $\alpha = 0.1$. However, the error associated with MNR-VEA was smaller than other methods, as evidenced by low ρ with smaller *a*.

3.5.6 Summary of 3 DOF

For all evaluated numerical methods (not including HMNR), the method required more computing time was MNR-VEA. The average ρ and maximum ρ were large, indicating solutions outside the domain. Homotopy was consistent in arriving at the closest root, and the choice of Δt did not markedly impact the results. HMNR was not able to give any solutions due to the nature of the method. MNR-VEA with $\alpha = 0.1$ has its highest performance in giving valid solutions (85% success); though, the maximum ρ value indicated the existence of solutions outside the domain (Table 3.16).

Table 3.16: Summary of the 3 DOF test case (17797 initial guesses).

Methods		CPU Time (s)	Percentage of initial guesses yielding valid solutions	Percentage of initial guesses yielding $\rho = 0$	ρ_{avg}(radians)	ρ_{max} (radians)
NR		35	87%	54%	2.0949×10^{11}	1.7205×10^{14}
Homotopy (Δt)	0.2	32	87%	81%	3.5995×10^{11}	1.4555×10^{14}
	0.1	42	87%	83%	4.5332×10^{11}	1.6260×10^{14}
	0.02	127	93%	84%	8.6112×10^{11}	7.6354×10^{13}
	0.01	231	95%	84%	6.7130×10^{11}	4.8780×10^{13}
HMNR (r)	2					
	3					
	4					
	5					
MNR-VEA (α)	0.1	1165	85%	3%	29.2098	14647.5
	0.01	3608	48%	11%	1.97745	567.754
	0.001	3360	51%	18%	0.52514	23.6008
	0.0001	3156	50%	17%	0.55197	86.0349

3.5.7 Converting a 3 DOF to a 1 DOF Problem

Performances of numerical methods for the 3 DOF test case did not necessarily meet expectations. Similarly to the 2 DOF test case, the 3 DOF problem can be converted into a 1

DOF test case. However, as shown in Appendix D, the 3 DOF problem gave unique analytical solutions during the conversion. Hence, a 3-to-1 test case was not needed or presented.

3.6 Variations and Computation Time

Alternate ways to evaluate and assess the robustness of an algorithm are by varying the input parameters, such as initial guesses, and by measuring the computation time. Assessment were made based on the following alteration, increase the initial guess by 1 degree or 1 unit for each case and find the average computation time by running it 10 times.

A summary of the results for 1 DOF are shown in Table 3.17. All numerical methods took less than 0.3 seconds to compute. NR and HMNR (r = 3,4,5) were the fastest; however, MNR-VEA (α = 0.01, 0.001, 0.0001) were the ones with highest performance in giving closest root (100% success).

A summary of the results for 2 DOF are shown in Table 3.18. Similar to 1 DOF, NR required the least computation time whereas Homotopy (with Δt = 0.01) took the longest to compute. MNR-VEA was consistent in giving valid solution (94% success) and has the highest successful rate of arriving at the closest root(between 44-46%). Overall, MNR-VEA with α = 0.0001 had the best performance.

A summary of the results for 3 DOF are shown in Table 3.19. HMNR was not able to give any solutions due to the nature of the method. NR and Homotopy had the best performance in terms of giving valid solution (100% success). Homotopy (Δt = 0.02, 0.01) had the highest successful rate of arriving at the closest root (95% success).

Table 3.17: Summary for 1 DOF average computing time (10 times) using 22 initial guesses

Methods		Average CPU time (s)	Percentage of initial guess yielding closest root
NR		0.020	68%
Homotopy	$\Delta t = 0.2$	0.022	41%
	$\Delta t = 0.1$	0.026	41%
	$\Delta t = 0.02$	0.053	55%
	$\Delta t = 0.01$	0.093	59%
HMNR	$r = 2$	0.226	73%
	$r = 3$	0.020	91%
	$r = 4$	0.020	82%
	$r = 5$	0.020	86%
MNR-VEA	$\alpha = 0.1$	0.027	86%
	$\alpha = 0.01$	0.026	100%
	$\alpha = 0.001$	0.027	100%
	$\alpha = 0.0001$	0.027	100%

Table 3.18: Summary for 2 DOF average computing time (10 times) using 1369 initial guesses

Methods		Average CPU time (s)	Percentage of initial guesses giving valid solution	Percentage of initial guess yielding $\rho = 0$
NR		1.886	97%	31%
Homotopy	$\Delta t = 0.2$	14.037	95%	40%
	$\Delta t = 0.1$	27.169	96%	41%
	$\Delta t = 0.02$	134.883	97%	41%
	$\Delta t = 0.01$	274.041	98%	40%
HMNR	$r = 2$	2.916	27%	15%
	$r = 3$	6.061	17%	11%
	$r = 4$	6.571	13%	9%
	$r = 5$	10.514	11%	9%
MNR-VEA	$\alpha = 0.1$	2.584	94%	45%
	$\alpha = 0.01$	2.875	94%	44%
	$\alpha = 0.001$	3.012	94%	45%
	$\alpha = 0.0001$	2.995	94%	46%

Table 3.19: Summary for 3 DOF average computing time (10 times) using 17797 initial guesses

Methods		Average CPU time (s)	Percentage of initial guesses giving valid solution	Percentage of initial guess yielding ρ =0
NR		36	100%	58%
Homotopy	Δt = 0.2	33	100%	85%
	Δt = 0.1	43	100%	89%
	Δt= 0.02	127	100%	95%
	Δt = 0.01	235	100%	95%
HMNR	r = 2			
	r = 3			
	r = 4			
	r = 5			
MNR-VEA	α = 0.1	1278	98%	18%
	α = 0.01	3684	55%	43%
	α = 0.001	3378	58%	49%
	α = 0.0001	3195	58%	48%

4 Discussion

4.1 Summary

In this study various numerical methods and test cases were evaluated to solve systems of non-linear equations.

For the 1 DOF test case, MNR-VEA (with α of 0.0001) and Homotopy did not fail to converge. Also, success with Homotopy in the root-finding process was independent of the value of Δt; though, Homotopy did require many more iterations to find a solution (5 to 20 times more iterations than MNR-VEA). NR, HMNR and some cases of MNR-VEA (with α values of 0.1, 0.01 and 0.001) failed to converge on closest root specifically when the initial guesses were in the region close to the critical point of 1.5708.

For the 2 DOF test case, MNR-VEA had a better performance in converging to the closest root, and was more consistent in obtaining valid solutions. Homotopy was consistent in arriving at the closest root, and more frequently arrived at the closest root as Δt decreased; though, the number of required iterations was large. HMNR gave the worst performance out of the four methods (and even resulted in complex numbers), and the choice of r was crucial in HMNR.

For the 2/1 DOF test case, MNR-VEA has a best performance in achieving x^* and x^{**}. MNR-VEA. These results, combined with the 1 DOF analysis, indicated that MNR-VEA is the best choice when dealing with 1 DOF problems.

For the 3 DOF test case, Homotopy was consistent in arriving at the closest root, and more frequently arrived at the closest root as Δt decreased. With Homotopy though, as well as NR,

average ρ and maximum ρ were large, indicating solutions outside the domain. MNR-VEA with α = 0.1 had its best performance in obtaining valid results converging to the closest root, with low average ρ. Though, solutions with ρ = 0 were minimal. HMNR was not able to solve the 3 DOF test case due to division by zero.

Computation time and percentage of giving closest root were measured to evaluate the performance of each numerical method. MNR-VEA is recommended for 1 and 2 DOF, and Homotopy (Δt = 0.02, 0.01) is recommended for 3 DOF.

Overall, the results of this analysis indicate different results for MNR-VEA and Homotopy. Specifically, MNR-VEA worked best with 1 and 2 DOF test cases (as well as the 2/1 DOF test case) whereas Homotopy worked moderately well with 1 and 2 DOF test cases and best with the 3 DOF test case. If the goal of the optimization is a mixture of both accuracy and minimal number of iterations, and the initial guess is approximately near the closest root, MNR-VEA may be more desirable to use for 1 and 2 DOF scenarios. Further, it is important to note that Homotopy has two parameters to be chosen: the auxiliary function, $g(x)$, and time increment, Δt. For Homotopy, there are rules to follow in picking $g(x)$ and Δt, and, even so, it does not guarantee convergence. As such, MNR-VEA may be more desirable to use since the choice of picking parameter r and relaxation parameter α is more intuitive (i.e., $\alpha < 1$).

4.2 Limitations

According to the results, here are some of the limitations:

- Homotopy, HMNR and MNR-VEA are modified versions of the Newton's method (NR). Even though each method is trying to improve on NR, they do not appear to avoid the problem of converging to a solution when the initial guess is close or at a critical point;
- For the 1 DOF analysis, limited numbers of initial guesses, which were not evenly spaced, were used to explore the behavior of each method. It would be beneficial to repeat this analysis with an alternate 1 DOF test case and employ even spacing.

5 Conclusions and Future Directions

5.1 Conclusions

After examining the three test cases with different numerical methods, the following conclusions can be drawn for 1 and 2 DOF test cases:

(1) MNR-VEA is most stable;

(2) Homotopy is able to give the closest root; however, an auxiliary function is needed and this may be difficult to select.

(3) NR and HMNR are not recommended since they perform in a similar manner, and they behaved poorly when the initial guess is closed to a critical point;

With regards to the higher (≥ 3) DOF test cases, the following conclusions can be drawn:

(1) Either NR or Homotopy (note, auxiliary functions are needed for Homotopy) are recommended. Their performances were the best out of all the numerical methods.

(2) HMNR had the worst performance since it gave the least number of closest root and it almost gave complex roots.

(3) MNR-VEA performance was unpredictable, and small α did not give better results.

5.2 Future Directions

For Homotopy, it would be prudent to apply different combinations of auxiliary functions, g, and Δt, to provide more insight in Homotopy's performance. For MNR-VEA, it would be prudent to try a larger range of relaxation parameter where $\alpha < 1$. It would also be useful to identify a

different 3 DOF problem which does not have an analytical solution to investigate MNR-VEA's proficiency with a "3-to-1" test case.

References

1. Kaw, A., Kalu, E.E., *Numerical Methods with Applications: Abridged*. 2011, University of South Florida.
2. Shmakov, S.L., *A Universal Method of Solving Quartic Equations.* International Jorunal of Pure and Applied Mathematics, 2011. **71**(2): p. 251-259.
3. Zoladek, H., *The Topological Proof of Abel-Ruffini Theorem.* Topological Methods in Nonlinear Analysis, 2000. **16**: p. 253-265.
4. He, J., *A modified Newton-Raphson method.* Communications in Numerical Methods in Engineering, 2004. **20**(10): p. 801-805.
5. Wu, T.M., *A study of convergence on the Newton-homotopy continuation method.* Applied Mathematics and Computation, 2005. **168**: p. 1169-1174.
6. Kumar, S., Sukavanam, N., Balasubramanian, R., *An Optimization Approach to Solve the Inverse Kinematics of Redundant Manipulator.* International Jorunal of Information and Systems Sciences, 2010. **6**(4): p. 414-423.
7. Tolani, D., Goswami, A., Badler, N.I., *Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs.* Graphical Models, 2000. **62**: p. 353-388.
8. Di Rocco, S., Eklund, D., Sommese, A., Wampler, C.W., *Algebraic C* - Actions and the Inverse Kinematics of a General 6R Mainipulator.* Applied Mathematics and Computation, 2010. **216**(9): p. 2512-2524.
9. Steele, J.A., *Some results concerning the fundamental nature of Wynn's vector epsilon algorithm,* in *Mechanical Engineering*. 2001, University of Saskatchewan: Saskatoon. p. 153.

10. Kalavrezos, M., Wennermo, M.,, *Stochastic Volatility Models in Option Pricing*, in *Department of Mathematics and Physics*. 2007, Malardalen University: Sweden. p. 104.
11. Kelley, C.T., *Solving nonlinear equations with Newton's Method*. 2003, Philadelphia: Society for Industrial and Applied Mathematics.
12. Moler, C., *MATLAB*. 2010, MathWorks: Natick, MA.
13. Kelley, C.T., *Iterative Methods for Linear and Nonlinear Equations*. 1995, Philadelphia: SIAM.
14. Galantai, A., *The theory of Newton's method.* Journal of Computional and Applied Mathematics, December 2000. **124**(1-2): p. 25-44.
15. Burden, R.L., Faires, J.D., *Numerical Analysis*. 9th ed. 2011: Brooks/Cole.
16. Epperson, J.F., *An introduction to numerical methods and analysis*. 2nd ed. 2013: Wiley.
17. Brent, R., *Algorithms for Minimization Without Derivatives*. 1973: Prentice-Hall.
18. Broyden, C.G., *A class of methods for solving nonlinear simultaneous equations.* Mathematics of Computation, 1965. **19**: p. 577-593.
19. Dembo, R.S., Eisentat, S.C., and Steihaug, T., *Inexact Newton methods.* SIAM Journal on Numerical Analysis, 1982. **19**: p. 400-408.
20. Eisenstat, S.C., Walker, H.F., *Choosing the Forcing Terms in an Inexact Newton Method.* SIAM Journal on Scientific Computing, 1996. **17**: p. 16-32.
21. Conn. A.R., G., N.M., Philippe, L.T., *Trust-Region Methods*. 2000: SIAM.
22. Nocedal, J., Wright, S.J., *Numerical Optimization*. 2nd ed. Springer Series in Operation Research. 2006: Springer.
23. Powell, M.J.D., *A FORTRAN subroutine for solving systems of nonlinear algebraic equations*. 1968, Harwell (England): Atomic Energy Research Establishment.

24. Levenberg, K., *A Method for the Solution of Certain Non-Linear Problems in Least Squares.* Quarterly of Applied Mathematics, 1944. **2**: p. 164-168.

25. Marquardt, D., *An Algorithm for Least-Squares Estimation of Nonlinear Parameters.* SIAM Journal on Applied Mathematics, 1963. **11**(2): p. 431-441.

26. Wynn, P., *Acceleration techniques for iterated vector and matrix problems.* Mathematics of Computation, 1962. **16**: p. 301-322.

27. Gekeler, E., *On the solution of systems of equations by the epsilon algorithm of Wynn.* Mathematics of Computation, April 1972. **26**(118): p. 427-436.

28. Brezinski, C., *Some results in the theiry of the vector e-algorithm.* Linear Algebra and its Applications, 1974. **8**: p. 77-86.

29. Sidi, A., *Application of vector extrapolation methods to consistent singular linear systems.* Applied Numerical Mathematics, 1989/90: p. 487-500.

30. Brezinski, C., *Application de l'e-algorithm a la resolution des systemes non lineaires.* Comptes Rendus Hebdomadaire des Seances de l'Academie des Sciences Paris, 1970. **Serie A**(271 A): p. 1174-1177.

31. Brezinski, C., Redivo-Zaglia, M., *The simplified topological e-algorithm: software and applications.* Numerical Algorithms, 2017. **74**(4): p. 1237-1260.

32. Bornemann, F., Laurie, D., Wagon, S., Waldvogel, J., *The SIAM 100-Digit Challenge*. 2004: SIAM.

33. Hafez, M.M., Cheng, H.K., *An acceleration technique related to Wynn's algorithm with application to transonic flow calculations.* Proceedings of the 5th International Conference on Numerical Methods in Fluid Dynamics, 1976.

34. Hafez, M.M., Palaniswamy, S., Kuruvila, G., Salas, M.D., *Applications of Wynn's e-algorithm to transoinc flow calculation.* AIAA Paper, 1987.

35. Cheung, S., Cheer, A., Hafez, M., Flores, J., *Convergence acceleration of viscous and inviscid hypersonic flow calculations.* AIAA Journal, 1991. **29**: p. 1214-1222.

36. Spong, M.W., Hutchinson, S., Vidyasagar, M., *Robot Modeling and Control*. 1 ed. 2005: Wiley.

37. Cai, H., Li, S., Hu, W., *A subsection algorithm for the inverse kinematics of the manipulators based on the simplex method.* 2006 6th International Conference on Intelligent Systems Design and Applications, 2006: p. 6.

38. Lenarcic, J., *An efficient numerical approach for calculating the inverse kinematics for robot manipulators.* Robotica, 1985. **3**(1): p. 21-26.

39. Hestenes, M.R., Stiefel, E., *Methods of Conjugate Gradients for Solving Linear Systems.* Journal of Research of the National Bureau of Standards, 1952. **49**(6): p. 409-436.

40. Caccavale, F., Wroblewski, W., *Comparsion of Newton-Raphson and Jacobian transpose inverse kinematics algorithms.* Proceedings of the Fifth International Symposium on Methods and Models in Automation and Robotics, 1998. **3**: p. 905-910.

41. Buss, S.R., *Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods*, in *Typeset manuscript*. April 2004: http://math.ucsd.edu/~sbuss/ResearchWeb.

42. Tagawa, K., Ohara, F., Ohta, Y., Haneda, H., *Direct inverse kinematics using fast interval bisection method and its automatic programming.* Proceedings of 1999 International Conference on Advanced Robotics, 1999: p. 497-502.

43. Cai, B., Zhang, Y., *Different-level redundancy-resolution and its equivalent relationship analysis for robot manipulators using gradient-descent and Zhang's neural-dynamic methods.* IEEE Transactions on Industrial Electronics, 2012. **59**(8): p. 3146-3155.

44. Martin, H.J.A., de Lope, J., Santos, M., *A method to learn the inverse kinematics of multi-link robots by evolving neuro-controllers.* Neurocomputing, 2009. **72**(13): p. 2806-2814.

45. Chu, I.P., Wu, T.M., *The kinematics design with the secant-homotopy continuation method.* Journal of the Chinese Society of Mechanical Engineers, 2010. **31**(2): p. 155-159.

46. Ren, J.L., Zheng, Z.B., Jiao, Z.M., *Simulation of virtual human running based on inverse kinematics.* ICETC 2010 - 2010 2nd International Conference on Education Technology and Computer, 2010: p. V3360-V3363.

47. Olsen, A.L., Petersen, H.G., *Inverse kinematics by numerical and analytical cyclic coordinate descent.* Robotica, 2011. **29**(4): p. 619-626.

48. Kreyszig, E., *Advanced Engineering Mathematics*. 9th ed. 2006: Wiley.

49. Bareiss, E.H., *Multistep Integer-Preserving Gaussian Elimination.* Argonne National Laboratory Report, May 1966. **ANL-7213**.

50. Brooks, R.M., Schmitt, K., *The contraction mapping principle and some applications.* Electronic Journal of Differential Equations, 2009: p. 1-90.

51. Adejumobi, I.A., Adepoju, G.A., Hamzat K.A., *Iterative Techniques for Load Flow Study: A Comparative Study for Nigerian 330kv Grid System as a Case Study* International Jorunal of Engineering and Advanced Technology, 2013. **3**(1): p. 153-158.

52. Hamouda, A., Zehar, K., *Improved algorithm for radial distribution networks load flow solution.* Electrical Power and Energy Systems, 2011. **33**: p. 508-514.

53. Lee, E., Mavroidis, C., *Solving the geometric design problem of spatial 3R robot manipulators using polynomial homotopy continuation.* Journal of Mechanical Design, 2002. **124**(4): p. 652-661.

54. Wu, T.M., *The inverse kinematics problem of spatial 4P3R robot manipulator by the homotopy continuation method with an adjustable auxiliary homotopy function.* Nonlinear Analysis, 2006. **64**(10): p. 2373-2380.

55. Floudas, C.A., Aggarwal, A., Ciric, A.R., *Global optimum search for nonconvex NLP and MINLP problems.* Computers & Chemical Engineering, 1989. **13**(10): p. 1117-1132.

56. Floudas, C.A., *Recent Advances in Global Optimization for Process Synthesis, Design and Control: Enclosure of All Solutions.* Computers & Chemical Engineering, 1999. **23**: p. S963-S973.

57. Gritton, K.S., Seader, J.D., Lin, W.J., *Global homotopy continuation procedures for seeking all roots of a nonlinear equation.* Computers & Chemical Engineering, 2001. **25**: p. 1003-1019.

58. Green, M.M., Wilson, A.N., *An Algorithm for Identifying Unstable Operating Points Using SPICE.* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1995. **14**(3): p. 360-370.

59. Yamamura, K., Kawata, H., *Interval solution of nonlinear equations using linear programming.* BIT Numerical Mathematics, 1998. **38**: p. 186-199.

60. Yamamura, K., Suda, K., *An efficient algorithm for finding all solutions of separable systems of nonlinear equations.* BIT Numerical Mathematics, 2007. **47**: p. 681-691.

61. Yamamura, K., Tamura, N., *Finding all solutions of separable systems of piecewise-linear equations using interger programming.* Journal of Computional and Applied Mathematics, 2012. **236**: p. 2844-2852.

62. Cajori, F., *Historical Note in the Newton-Raphson Method of Approximation.* The American Mathematical Monthly, 1911. **18**(2): p. 29-32.

63. Deuflhard, P., *A Short History of Newton's Method.* Documenta Mathematica, 2012. **Extra**: p. 25-30.

64. Deuflhard, P., *Newton Methods for Nonlinear Prolems: Affine Invariance and Adaptive Algorithms*. 2004, New York: Springer.

65. Simons, S., *A Modification of the Newton-Raphson Method.* The Mathematical Gazette, 2006. **90**(517): p. 128-130.

66. Li J., L.L., Jiang Y., *An Accelerating Method of Training Neural Networks Based On Vector Epsilon Algorithm.* Proceedings of the Third International Conference on Information and Computing Science, 2010: p. 292-295.

67. Wynn, P., *On a device for calculating the $e_m(S_n)$ transformations.* Mathematical Tables and Other Aids to Computation, 1956. **10**: p. 91-96.

68. Shanks, D., *Non-linear transformations of divergent and slowly convergent sequence.* Journal of Mathematics and Physics, 1955. **34**: p. 1-42.

69. Schmidt, R.J., *On the numerical solution of linear simultaneous equations by an iterative method.* Philosophy Magazine, 1941. **32**(7): p. 369-383.

70. Dolovich, A.T., Brodland, G.W., *Convergence acceleration for iterative finite-element methods.* Journal of Engineering Mechaincs, 1995. **121**: p. 1-6.

71. Lowe, D.S., *Enhanced Convergence of Algebraic Reconstruction Algorithms Used in Computerised Tomography*, in *Mechanical Engineering*. 1996, University of Saskatchewan: Saskatoon. p. 117.

72. Barrett, R., Berry, M., Cham, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., van der Vorst, H., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. 1994: SIAM.

73. Wu, T.M., *Searching All the Roots of Inverse Kinematics Problem of Robot by Homotopy Continuation Method.* Journal of Applied Sciences, 2005. **5**(4): p. 666-673.

Appendix A–Verification of the limit of partial sum of a geometric series and the Scalar Epsilon Algorithm

The goal of this Appendix is to show that the limit of partial sum of a geometric series x^*, calculated using Shanks Transform, is identical to the transform value C from SEA.

Using equation (2.62) to calculate the limit, x^*, of Shanks Transform for any three consecutive iterates x_m, x_{m+1}, and x_{m+2}.

$$x^* = \frac{\begin{vmatrix} x_m & \Delta x_m \\ x_{m+1} & \Delta x_{m+1} \end{vmatrix}}{\begin{vmatrix} 1 & \Delta x_m \\ 1 & \Delta x_{m+1} \end{vmatrix}}, \tag{2.62}$$

where $\Delta x_m = x_{m+1} - x_m$, $\Delta x_{m+1} = x_{m+2} - x_{m+1}$.

$$x^* = \frac{\begin{vmatrix} x_m & x_{m+1} - x_m \\ x_{m+1} & x_{m+2} - x_{m+1} \end{vmatrix}}{\begin{vmatrix} 1 & x_{m+1} - x_m \\ 1 & x_{m+2} - x_{m+1} \end{vmatrix}}, \tag{A.1}$$

$$x^* = \frac{(x_m)(x_{m+2} - x_{m+1}) - (x_{m+1} - x_m)(x_{m+1})}{(x_{m+2} - x_{m+1}) - (x_{m+1} - x_m)}, \tag{A.2}$$

$$x^* = \frac{x_m x_{m+2} - x_m x_{m+1} - x_{m+1} x_{m+1} + x_{m+1} x_m}{x_{m+2} - x_{m+1} - x_{m+1} + x_m}, \tag{A.3}$$

$$x^* = \frac{x_m x_{m+2} - x_{m+1} x_{m+1}}{x_{m+2} - 2x_{m+1} + x_m}. \tag{A.4}$$

Table A .1: SEA Example

Col. 1	Col. 2	Col. 3	Col. 4
0			
	x_m		
0		A	
	x_{m+1}		C
0		B	
	x_{m+2}		
0			

From Table A.1,

$$\mathrm{A} = 0 + \frac{1}{x_{m+1} - x_m}, \tag{A.5}$$

$$\mathrm{B} = 0 + \frac{1}{x_{m+2} - x_{m+1}}, \tag{A.6}$$

$$\mathrm{C} = x_{m+1} + \frac{1}{\mathrm{B} - \mathrm{A}}. \tag{A.7}$$

Substituting equations (A.5) and (A.6) into equation (A.7),

$$\mathrm{C} = x_{m+1} + \frac{1}{\dfrac{1}{x_{m+2} - x_{m+1}} - \dfrac{1}{x_{m+1} - x_m}}, \tag{A.8}$$

$$\mathrm{C} = x_{m+1} + \frac{1}{\dfrac{x_{m+1} - x_m}{\left(x_{m+2} - x_{m+1}\right)\left(x_{m+1} - x_m\right)} - \dfrac{x_{m+2} - x_{m+1}}{\left(x_{m+1} - x_m\right)\left(x_{m+2} - x_{m+1}\right)}}, \tag{A.9}$$

$$\mathrm{C} = x_{m+1} + \frac{1}{\dfrac{x_{m+1} - x_m - x_{m+2} + x_{m+1}}{x_{m+2}x_{m+1} - x_{m+2}x_m - x_{m+1}x_{m+1} + x_{m+1}x_m}}, \tag{A.10}$$

$$\mathrm{C} = x_{m+1} + \frac{x_{m+2}x_{m+1} - x_{m+2}x_m - x_{m+1}x_{m+1} + x_{m+1}x_m}{x_{m+1} - x_m - x_{m+2} + x_{m+1}}, \tag{A.11}$$

$$C = x_{m+1} + \frac{x_{m+2}x_{m+1} - x_{m+2}x_m - x_{m+1}x_{m+1} + x_{m+1}x_m}{-x_{m+2} + 2x_{m+1} - x_m}, \tag{A.12}$$

$$C = \frac{(x_{m+1})(-x_{m+2} + 2x_{m+1} - x_m)}{-x_{m+2} + 2x_{m+1} - x_m} + \frac{x_{m+2}x_{m+1} - x_{m+2}x_m - x_{m+1}x_{m+1} + x_{m+1}x_m}{-x_{m+2} + 2x_{m+1} - x_m}, \tag{A.13}$$

$$C = \frac{-x_{m+1}x_{m+2} + 2x_{m+1}x_{m+1} - x_{m+1}x_m + x_{m+2}x_{m+1} - x_{m+2}x_m - x_{m+1}x_{m+1} + x_{m+1}x_m}{-x_{m+2} + 2x_{m+1} - x_m}, \tag{A.14}$$

$$C = \frac{x_{m+1}x_{m+1} - x_{m+2}x_m}{-x_{m+2} + 2x_{m+1} - x_m}, \tag{A.15}$$

$$C = \frac{x_m x_{m+2} - x_{m+1}x_{m+1}}{x_{m+2} - 2x_{m+1} + x_m}. \tag{A.16}$$

Appendix B – Moore-Penrose Inverse's properties

This appendix summarizes the properties of Moore-Penrose Inverse of matrix **A**, where **A** is a non-singular matrix.

Given the size of matrix **A** is $\mathrm{m} \times \mathrm{n}$.

1. If m < n, then right inverse, $\mathbf{A}^{\mathrm{R}}$, exists;

2. If m > n, then left inverse, $\mathbf{A}^{\mathrm{L}}$, exists.

$$\mathbf{A}\,\mathbf{A}^{\mathrm{R}} = \mathbf{I} \tag{B.1}$$

$$\mathbf{A}^{\mathrm{L}}\mathbf{A} = \mathbf{I} \tag{B.2}$$

Find the left inverse, $\mathbf{A}^{\mathbf{L}}$, of **A**, e.g.

$$\mathbf{A} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \tag{B.3}$$

$$\mathbf{A}^{\mathrm{T}}\mathbf{A} = \begin{bmatrix} a_1 & a_2 & \cdots & a_m \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \tag{B.4}$$

$$\mathbf{A}^{\mathrm{T}}\mathbf{A} = a_1^2 + a_1^2 + \cdots + a_m^2 \tag{B.5}$$

$$\mathbf{A}^{\mathrm{T}}\mathbf{A} = \|\mathbf{A}\|^2 \tag{B.6}$$

Let

$$\mathbf{B} = \frac{\mathbf{A}^{\mathbf{T}}}{\|\mathbf{A}\|^2} \tag{B.7}$$

Then

$$\mathbf{B}\,\mathbf{A} = \frac{\mathbf{A}^{\mathbf{T}}}{\|\mathbf{A}\|^2}\,\mathbf{A} \tag{B.8}$$

$$\mathbf{B}\,\mathbf{A} = \frac{\mathbf{A}^{\mathbf{T}}\mathbf{A}}{\|\mathbf{A}\|^2} = \frac{\|\mathbf{A}\|^2}{\|\mathbf{A}\|^2} = 1 \tag{B.9}$$

Therefore,

$$\mathbf{A}^{\mathrm{L}} = \mathbf{B} = \frac{\mathbf{A}^{\mathrm{T}}}{\|\mathbf{A}\|^2} \tag{B.10}$$

Finding the right inverse, $\mathbf{A}^{\mathbf{R}}$, of **A**, e.g.

$$\mathbf{A} = \begin{bmatrix} a_1 & a_2 & \cdots & a_m \end{bmatrix} \tag{B.11}$$

$$\mathbf{A}\,\mathbf{A}^{\mathbf{T}} = \begin{bmatrix} a_1 & a_2 & \cdots & a_m \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \tag{B.12}$$

$$\mathbf{A}\,\mathbf{A}^{\mathbf{T}} = a_1^2 + a_1^2 + \cdots + a_m^2 \tag{B.13}$$

$$\mathbf{A}\,\mathbf{A}^{\mathbf{T}} = \|\mathbf{A}\|^2 \tag{B.14}$$

Let

$$\mathbf{B} = \frac{\mathbf{A}^{\mathbf{T}}}{\|\mathbf{A}\|^2} \tag{B.15}$$

Then

$$\mathbf{A}\,\mathbf{B} = \mathbf{A}\frac{\mathbf{A}^{\mathbf{T}}}{\|\mathbf{A}\|^2} \tag{B.16}$$

$$\mathbf{A}\,\mathbf{B} = \frac{\mathbf{A}\,\mathbf{A}^{\mathbf{T}}}{\|\mathbf{A}\|^2} = \frac{\|\mathbf{A}\|^2}{\|\mathbf{A}\|^2} = 1 \tag{B.17}$$

Therefore,

$$\mathbf{A}^{\mathrm{R}} = \mathbf{B} = \frac{\mathbf{A}^{\mathrm{T}}}{\|\mathbf{A}\|^2} \tag{B.18}$$

Moore-Penrose Inverse of **A**, **B** must satisfy the following:

(i) $$\mathbf{A}\,\mathbf{B}\,\mathbf{A} = \mathbf{A} \tag{B.19}$$

(ii) $$\mathbf{B}\,\mathbf{A}\,\mathbf{B} = \mathbf{B} \tag{B.20}$$

(iii) $$(\mathbf{A}\,\mathbf{B})^{\mathbf{T}} = \mathbf{A}\,\mathbf{B} \tag{B.21}$$

(iv) $$(\mathbf{B}\,\mathbf{A})^{\mathbf{T}} = \mathbf{B}\,\mathbf{A} \tag{B.22}$$

Property (i) $\mathbf{A\,B\,A} = \mathbf{A}$, for $\mathbf{A} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}$

$$\mathbf{A\,B\,A} = \frac{\mathbf{A\,A^T}}{\|\mathbf{A}\|^2} = \frac{1}{\|\mathbf{A}\|^2} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \begin{bmatrix} a_1 & a_2 & \cdots & a_m \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \tag{B.23}$$

$$\mathbf{A\,B\,A} = \frac{1}{\|\mathbf{A}\|^2} \begin{bmatrix} a_1^2 & a_1 a_2 & \cdots & a_1 a_m \\ a_2 a_1 & \ddots & & \\ \vdots & & \ddots & \\ a_m a_1 & & & a_m^2 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \tag{B.24}$$

$$\mathbf{A\,B\,A} = \frac{1}{\|\mathbf{A}\|^2} \begin{bmatrix} a_1^3 + a_1 a_2^2 + \cdots + a_1 a_m^2 \\ a_2 a_1^2 + a_2^3 + \cdots + a_2 a_m^2 \\ \vdots \\ a_m a_1^2 + a_m a_2^2 + \cdots + a_m^3 \end{bmatrix} \tag{B.25}$$

$$\mathbf{A\,B\,A} = \frac{1}{\|\mathbf{A}\|^2} \begin{bmatrix} a_1 \left(a_1^2 + a_2^2 + \cdots + a_m^2 \right) \\ a_2 \left(a_1^2 + a_2^2 + \cdots + a_m^2 \right) \\ \vdots \\ a_m \left(a_1^2 + a_2^2 + \cdots + a_m^2 \right) \end{bmatrix} \tag{B.26}$$

$$\mathbf{A\,B\,A} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = \mathbf{A} \tag{B.27}$$

Property (ii) $\mathbf{B\,A\,B} = \mathbf{B}$, for $\mathbf{A} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}$

$$\mathbf{B}\,\mathbf{A}\,\mathbf{B} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = \mathbf{B} \tag{B.28}$$

Appendix C – Converting 2 DOF to 1 DOF

This is the derivation for converting a 2 DOF problem in Section 3.5 to a 1 DOF problem.

$$\mathrm{L}_1 \cos\beta + \mathrm{L}_2 \cos\gamma - x_\mathrm{P} = 0, \tag{3.8}$$

$$\mathrm{L}_1 \sin\beta + \mathrm{L}_2 \sin\gamma - y_\mathrm{P} = 0, \tag{3.9}$$

where $\mathrm{L}_1 = 2$, $\mathrm{L}_2 = 1$, and $(x_\mathrm{P}, y_\mathrm{P}) = (-1, 1.5)$. Rearranging equations (3.8) and (3.9) becomes

$$\cos\gamma = -1 - 2\cos\beta, \tag{C.1}$$

$$\sin\gamma = 1.5 - 2\sin\beta. \tag{C.2}$$

Squaring and adding equations (C.1) and (C.2) gives

$$\cos^2\gamma + \sin^2\gamma = (-1 - 2\cos\beta)^2 + (1.5 - 2\sin\beta)^2. \tag{C.3}$$

Since $\cos^2\gamma + \sin^2\gamma = 1$,

$$1 = 1 + 4\cos\beta + 4\cos^2\beta + 2.25 - 6\sin\beta + 4\sin^2\beta, \tag{C.4}$$

$$0 = 4\cos\beta - 6\sin\beta + 2.25 + 4(\cos^2\beta + \sin^2\beta). \tag{C.5}$$

Since $\cos^2\beta + \sin^2\beta = 1$,

$$4\cos\beta - 6\sin\beta + 6.25 = 0. \tag{C.6}$$

Appendix D – Analytical solution of the 3 DOF problem

This is the analytical solution of the 3 DOF problem in Section 3.5.

$$-\mathrm{L}_2 \cos\beta \sin\gamma + a - x_\mathrm{P} = 0, \tag{3.21}$$

$$\mathrm{L}_2 \cos\gamma + c + \mathrm{L}_1 - y_\mathrm{P} = 0, \tag{3.22}$$

$$\mathrm{L}_2 \sin\beta \sin\gamma - b - z_\mathrm{P} = 0, \tag{3.23}$$

Obtaining $\cos\gamma$ by rearranging equation (3.22)

$$\cos\gamma = \frac{y_\mathrm{P} - c - \mathrm{L}_1}{\mathrm{L}_2}, \tag{D.1}$$

Letting $\cos\gamma = \xi$

$$\xi = \frac{y_\mathrm{P} - c - \mathrm{L}_1}{\mathrm{L}_2}, \tag{D.2}$$

Therefore,

$$\cos^2\gamma = \xi^2, \tag{D.3}$$

Since $\cos^2\gamma + \sin^2\gamma = 1,$

$$\sin^2\gamma = 1 - \cos^2\gamma, \tag{D.4}$$

$$\sin^2\gamma = 1 - \xi^2, \tag{D.5}$$

The quantity $\cos^2\beta$ can be obtained by rearranging and squaring equation (3.21)

$$-\mathrm{L}_2 \cos\beta \sin\gamma = x_\mathrm{P} - a, \tag{D.6}$$

$$\mathrm{L}_2{}^2 \cos^2\beta \sin^2\gamma = (x_\mathrm{P} - a)^2, \tag{D.7}$$

$$\cos^2\beta = \frac{(x_\mathrm{P} - a)^2}{\mathrm{L}_2{}^2 \sin^2\gamma}, \tag{D.8}$$

$$\cos^2\beta = \frac{(x_P - a)^2}{L_2{}^2(1-\xi^2)}, \tag{D.9}$$

The quantity $\sin^2\beta$ can then be obtained by rearranging and squaring equation (3.23)

$$L_2 \sin\beta \sin\gamma = z_P + b, \tag{D.10}$$

$$L_2{}^2 \sin^2\beta \sin^2\gamma = (z_P + b)^2, \tag{D.11}$$

$$\sin^2\beta = \frac{(z_P + b)^2}{L_2{}^2 \sin^2\gamma}, \tag{D.12}$$

$$\sin^2\beta = \frac{(z_P + b)^2}{L_2{}^2(1-\xi^2)}, \tag{D.13}$$

Since $\cos^2\beta + \sin^2\beta = 1$, from equations (D.9) and (D.13)

$$\frac{(x_P - a)^2}{L_2{}^2(1-\xi^2)} + \frac{(z_P - b)^2}{L_2{}^2(1-\xi^2)} = 1, \tag{D.14}$$

$$(x_P - a)^2 + (z_P - b)^2 = L_2{}^2(1-\xi^2), \tag{D.15}$$

$$\xi = \pm\sqrt{1 - \frac{(x_P - a)^2 + (z_P - b)^2}{L_2{}^2}}\,. \tag{D.16}$$